Universität des Saarlandes, FR Informatik
Max-Planck-Institut für Informatik, AG 1

Exact Arrangement Computation for Cubic Curves

Diplomarbeit im Fach Informatik
Master’s Thesis in Computer Science

von / by

Arno Eigenwillig

unter Anleitung von / supervised by

Prof. Dr. Elmar Schömer

Zweitgutachter / second examiner

Prof. Dr. Kurt Mehlhorn

Saarbrücken, 30. Juli 2003

Meinen Eltern

Acknowledgements

First and foremost, I thank Professor Elmar Schömer for this interesting and fruitful topic and his continuous help and advice.

I want to express my special gratitude to Nicola Wolpert for her valuable comments on a draft version of this text, and for her support throughout the past year which has guided my work and made it much more enjoyable.

The EXACUS team members Eric Berberich, Michael Hemmer, Susan Hert, Lutz Kettner, Kurt Mehlhorn, Joachim Reichel and Susanne Schmitt have helped me in many ways, and I wish to thank all of them.

I am grateful for the scholarship I received from the German National Academic Foundation *(Studienstiftung des deutschen Volkes)* during my studies and the preparation of this thesis.

This work has partially been supported by the IST Programme of the European Union as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG – Effective Computational Geometry for Curves and Surfaces).

Hilfsmittelerklärung *(Non-plagiarism statement)*
Hiermit versichere ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Saarbrücken, den 30. Juli 2003,

(Arno Eigenwillig)

Zusammenfassung

Die vorliegende Arbeit beschreibt ein exaktes und vollständiges Verfahren zur Berechnung der Schnittfigur (des Arrangements) einer endlichen Menge von Segmenten ebener algebraischer Kurven vom Höchstgrad 3, das zugleich eine brauchbare Effizienz hat. Dazu wird eine verbesserte Fassung des Besenlinien-Algorithmus von Bentley und Ottmann (1979) mit geometrischen Grundoperationen unterlegt, die durch effiziente Techniken des symbolischen Rechnens mit Polynomen verwirklicht sind. Es liegt eine Umsetzung des Verfahrens in ein C++-Programm vor, und die Arbeit enthält Laufzeitmessungen.

Abstract

This thesis presents an exact and complete method to compute the arrangement of a finite set of segments of algebraic curves of degree up to 3 which has a practically feasible efficiency. For this, an improved version of the sweep line algorithm by Bentley and Ottmann (1979) is supplied with basic geometric operations that are realized by efficient techniques of symbolic computation with polynomials. There is a software implementation of this method written in C++, and the thesis contains running time measurements.

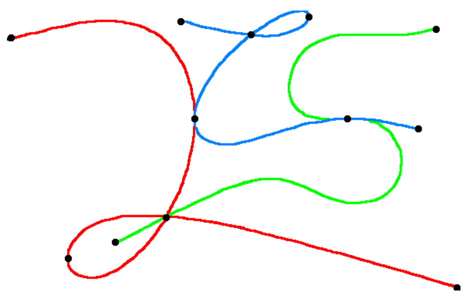
Contents

Chapter 1

Introduction

Problem Statement

A finite set of segments of planar curves induces a subdivision of the plane into vertices, edges, and faces, called the *arrangement* of the segments. Its vertices are the endpoints and intersection points of the segments, its edges are formed by the pieces of segments connecting the vertices, and its faces are the regions of the plane bounded by the edges. An arrangement can be represented as a graph with a cyclic ordering of the edges incident to each vertex, with vertices labelled by point coordinates and edges referring back to the segments they come from. Computing this graph is an important fundamental operation in computational geometry, for example it is the pivotal step in regularized boolean operations on polygons bounded by the segments at hand.


Segments of three cubic curves with singular and tangential intersections.

The central object of this thesis is to supply a generalized version of the Bentley-Ottmann sweep line algorithm for arrangement computation [BO79] with the necessary basic geometric operations to use it on segments of *cubic curves*, i. e. solution sets of equations $f(x,y) = 0$, where f is a polynomial of total degree at most three and has rational coefficients. This includes line segments, conic arcs, and cubic splines.

The method we present is *exact*, that is to say, the output represents the true mathematical result undistorted by rounding or the like, and *complete*, meaning that we allow all inputs, even highly degenerate ones (such as the segments shown above) which cannot be analyzed reliably with straightforward numerical computation. At the same time, our method is reasonably *efficient*. We have produced an implementation to provide evidence in favour of this and of the practical usability of our algorithm; it computes the arrangement of the curves supporting the segments depicted above in a fraction of a second. To our knowledge, we are the first to reach all three goals simultaneously. A short technical report on our results [ES$^+$02] appeared in December 2002.

Our method consists of the the actual sweep line procedure combined with pre- and post-processing steps necessary to cope with inherent restrictions of the sweep line approach (x-monotonicity of segments) and of the algebra used for the basic geometric operations (non-degenerate choice of coordinates). Altogether, we obtain an algorithm taking a set of segments, delimited by rational endpoints, to the graph representing their arrangement, with point coordinates given by floating-point approximations of the internal symbolic representation that obey an arbitrarily small user-defined error bound. As a variant, we offer arrangement computation for sets of entire cubic curves instead of segments.

Related Work

Computing arrangements, although mostly of linear objects, is a major focus in computational geometry; see the survey articles of Halperin [H97] and Agarwal and Sharir [AS00]. Many exact methods to handle curved objects have been formulated for the Real RAM model of computation [PS85], which allows constant-time operations on arbitrary algebraic numbers. This idealized model ignores the fact that exact computation with algebraic numbers is very costly. The idea of exact geometric computation proposed by Yap et al. [YD95] [YM01] [LP$^+$03] decouples the notions of arithmetic and geometric exactness.

There is a branch of computational geometry research trying to avoid explicit arithmetic with algebraic numbers at all and retain the notion from straight-line algorithmic geometry that predicate evaluation is sign determination of polynomial expressions in the input data. Devillers et al. [DF$^+$00] discuss predicates for arrangement computation for circles in this style, but their approach does not extend easily to more complicated curves.

The computer aided design (CAD) community has been addressing curve-curve intersections for a long time, but traditionally the focus is on fast approximate floating-point methods and not on exactness. MAPC [KC$^+$99] is a library for exact computation and manipulation of algebraic curves aimed at CAD purposes. It includes the computation of arrangements of planar curves but does not handle all degenerate situations.

Analyzing the topology of one real algebraic curve has also been a research topic in symbolic computation, see for example Sakkalis [S91] and Hong [H93].

In the computational geometry community, exact, efficient, and complete algorithms for planar arrangements have been developed by Wein [We02a] [We02b] and Berberich et al. [BE$^+$02] [H02] for conic segments, and by Wolpert [GH$^+$01] [W02] for special quartic curves as part of a surface intersection algorithm.

The method presented here builds on the approach of Berberich et al., including their variant of the Bentley-Ottmann algorithm. However, the number types involved make a major difference. Conics allow root expression arithmetic for their $y(x)$ parametrization and for all vertex coordinates except transversal intersections, facilitating the use of the `leda::real` [BFMS00] [MN99, 4.4] or `CORE::expr` [KL$^+$99] number types. This is not true for cubics, hence for them the basic geometric operations have to find different ways of striking a balance between exact arithmetic and efficiency, see below. The more complicated geometry of cubic curves has also called for the introduction of curve and curve pair analyses as a separate abstraction level below segments and their geometric predicates.

Parts of our approach to these geometric analyses stem from Wolpert's work [W02], but we use stripes instead of boxes, saving half of all resultant computations: We project the intersection points of two curves to their x-coordinates by computing the Sylvester resultant. When extending back from the projection into the plane, we try and avoid explicit arithmetic with the irrational x-coordinates by inspecting the curves at the rational boundaries of a stripe containing the intersection. Thus, we get by mostly with exact integer and rational arithmetic.

Outline

The text begins with two foundational chapters. The first, Chapter 2, surveys the **Algebraic Foundations** of our method and introduces the algorithmic tools we use from the domain of symbolic computation with polynomials, viz. computation of greatest common divisors, square-free factorization, (sub)resultants, and Uspensky's Method. Then it discusses several representations of roots of rational polynomials. We will mainly use the one from §2.4.4 whereby a root is represented as the polynomial and an interval with rational boundary points that contains this root but isolates it from all others.

Chapter 3 discusses the **Geometry of Algebraic Curves**, as far as necessary for our purposes. There are essentially four topics covered in it: First, basics of plane algebraic geometry like components, intersection multiplicity, and Bezout's Theorem. Second, how to use the Implicit Function Theorem to view a plane algebraic curve as consisting of arcs (maximal x-monotone pieces) that meet at x-extreme points and singularities. Third, a classification of cubic curves and questions of (ir)rationality of components and special points (since this governs their algorithmic representation). Finally, we look at the issue of converting rational parametric curves, in particular splines, to algebraic, i. e. implicit, curves.

A few less essential but interesting **Mathematical Addenda** to those two chapters are contained in Appendix A.

Chapter 4 contains the central part of our work: The **Algorithmic Analysis of Cubic Curves and Curve Pairs**, which is about determining the relative position of their arcs in y-direction as a function of x, including the detection of intersection points. Those analyses of curves and curve pairs introduce conditions on the choice of a coordinate system that avoids certain degeneracies. They are established probabilistically by a random change of coordinates before and checked during the computation.

After all, we come to **Arrangements of Segments of Algebraic Curves** in Chapter 5. We base an effective notion of sweepable segments on the analysis of curve pairs and apply the Bentley-Ottmann sweep line algorithm. That chapter also discusses the pre-processing of segments to make them sweepable.

Chapter 6 reports on the **Implementation** of our algorithm as part of the EXACUS project and on running time measurements. The text concludes with **Conclusions** in Chapter 7.

Chapter 2

Algebraic Foundations

This chapter surveys fundamental algebraic matters on which the subsequent chapters build.

Algebraic matters deserve to be our starting point, since the algorithmic handling of algebraic curves rests to a large extent on algebraic operations. The main reference for the abstract algebra presented here is the famous textbook by Lang [L84] (in English), or alternatively the textbook by Bosch [B96] (in German). The algorithmic aspects are covered in Geddes-Czapor-Labahn [GC$^+$92], von zur Gathen-Gerhard [GG99], and Cohen [C93].

2.1 Notation and Terminology

Recall the algebraic notions of ring and field. We will use them in the following restricted meaning: A *field* for our purposes always is a field of characteristic zero, i. e. a superset of the rational numbers $\mathbb{Q}$. Besides $\mathbb{Q}$ itself, this includes its various algebraic extensions, the real numbers $\mathbb{R}$, and the complex numbers $\mathbb{C} = \mathbb{R}(i)$.

A *ring* in modern terminology is a commutative ring with unity. We additionally demand it to contain the integers $\mathbb{Z}$, and most of the time, unless specifically stated otherwise, to be free of zero divisors. Examples of rings in this sense are $\mathbb{Z}$ and rings of polynomials over $\mathbb{Z}$ or over a field. The multiplicatively invertible elements of a ring R, the so-called *units* of R, form a multiplicative group, denoted R^*.

With our terminology, a ring R is a field iff every element besides zero is a unit, and every ring R possesses an essentially unique *quotient field* $\mathrm{Q}(R)$, i. e. a $\subseteq$-minimal field containing R as a subring. For example, $\mathrm{Q}(\mathbb{Z}) = \mathbb{Q}$.

Besides these exceptions, we strive to use standard terms and symbols, including the word “iff” to mean “if and only if”. Square brackets $[x]_\sim$ denote equivalence classes modulo an equivalence relation $\sim$.

2.2 Polynomials

2.2.1 Basics

Adjoining an indeterminate x to a ring R yields the *univariate* polynomial ring $R[x]$. For a polynomial $f \in R[x]$ we speak of its *degree* $\deg(f) = \deg_x(f)$, its *leading coefficient* $\ell(f) = \ell_x(f)$ (both of which are defined as 0 for $f = 0$), and its *leading monomial* $\ell(f)x^{\deg(f)}$. We call a polynomial *monic* if its leading coefficient is 1.

By repeating the adjunction of variables to a ring, we obtain rings of *multivariate* polynomials: $R[x_1]\cdots[x_n] = R[x_1,\dots,x_n]$. The order of adjunction does not matter. For $n = 2$ we call them *bivariate* polynomials.

A multivariate polynomial can either be seen *hierarchically* as a univariate polynomial in the outermost variable x_n with coefficients in $R[x_1]\cdots[x_{n-1}]$, or *flat* as a sum of its *monomials* $a_{i_1\dots i_n}x_1^{i_1}\cdots x_n^{i_n}$ with $a_{i_1\dots i_n} \neq 0$. The *total degree* $\deg(f)$ of a non-zero polynomial f is the highest sum of exponents $i_1 + \ldots + i_n$ among all of its monomials. Again, we define $\deg(0) = 0$. Polynomials of degree 0, 1, 2, 3, … are called *constant*, *linear*, *quadratic*, *cubic*, etc.

A third, *pointwise* view on polynomials stems from the maps they induce: Substituting $(x_1,\dots,x_i)$ by elements of some ring $S \supseteq R$ in a polynomial f yields the i-th *evaluation map*

$$\begin{aligned} R[x_1,\dots,x_n] \times S^i &\rightarrow S[x_{i+1},\dots,x_n] \\ (f,\xi) &\mapsto f|_\xi = f(\xi_1,\dots,\xi_i,x_{i+1},\dots,x_n) \end{aligned} \tag{2.1}$$

For fixed ξ, this is the *evaluation homomorphism* $f \mapsto f|_\xi$ at ξ. Fixing f instead yields the *polynomial map* $\xi \mapsto f|_\xi$ induced by f. Since rings, according to our convention, are infinite and possess quotient fields, a non-zero polynomial can only have finitely many zeroes (cf. §2.2.4), so that polynomials and polynomial maps correspond bijectively. Therefore, we can identify polynomials with the polynomial maps they induce.

A polynomial $f \in R[x_1,\dots,x_n]$ is called x_n*-regular* if it contains a monomial of the form $cx_n^{deg(f)}$ with a non-zero constant $c \in R$. In other words, f is x_n-regular iff its total degree $\deg(f)$ in the flat view and its univariate degree $\deg_{x_n}(f)$ in the hierarchical view are equal. This is a helpful property for considerations which mix the hierarchical and the flat view. Regularity also implies that both notions of degree remain invariant under evaluation in any subset of the inner variables (unless one substitutes polynomials for variables).

Let $f \in R[x_1,\dots,x_n]$ be non-zero. By sorting the reversed degree vectors $(i_n,\dots,i_1)$ of its monomials lexicographically, we can define the *leading monomial* and *leading coefficient* $\ell(f)$ of a multivariate polynomial f. We have chosen to reverse the degree vector because then an x_n-regular polynomial has the same leading coefficient in the flat and hierarchical views. The leading monomial of a product fg of polynomials is the product of the leading monomials of f and g, hence $\deg(fg) = \deg(f) + \deg(g)$.

Proposition 2.2.1: *Let R be a ring, and let $f \in R[x_1, \dots, x_n]$ be x_n-regular. Then any factor g of f is also x_n-regular.*

Proof: Let $f = gh$ with $g, h \in K[x_1, \dots, x_n]$. Then $\deg(f) = \deg(g) + \deg(h) \geq \deg_{x_n}(g) + \deg_{x_n}(h) = \deg_{x_n}(f) = \deg(f)$, hence equality holds throughout. It follows that $\deg(g) = \deg_{x_n}(g)$. □

2.2.2 Unique factorization and the GCD

Let K be a field. Then $K[x]$ is a *Euclidean ring* by virtue of division with remainder:

Proposition 2.2.2: *(Euclidean division of polynomials)*
Let K be a field. For any two polynomials $f, g \in K[x]$, $g \neq 0$, there exists a unique quotient q and remainder r in $K[x]$ such that

$$f = qg + r, \quad \deg(r) < \deg(g). \tag{2.2}$$

The constructive proof of this proposition yields a simple and reasonably efficient algorithm to compute these quantities. Iff g divides f, which we denote by $g|f$, then Euclidean division produces the remainder 0 and the quotient f/g.

Euclidean division allows us to compute greatest common divisors. Recall that in any ring R a *greatest common divisor* (gcd) of two elements $r, s \in R$ is an element $d \in R$ such that $d|r \wedge d|s$, and $d'|r \wedge d'|s \Rightarrow d'|d$ for all $d' \in R$. (This means a gcd is a greatest lower bound under the preorder relation of divisibility.)

To compute a gcd in $K[x]$, we apply the *Euclidean Algorithm*: Given two non-zero polynomials $f, g \in K[x]$, repeatedly replace one with larger (or equal) degree with its remainder modulo the other one, until zero occurs as remainder (which must eventually happen because the degrees of remainders form a strictly decreasing sequence of natural numbers). Then the last non-zero remainder d is a greatest common divisor of f and g. By accumulating intermediate results, the *Extended Euclidean Algorithm* additionally computes *Bezout factors* $p, q \in K[x]$ with the property $d = pf + qg$. See [GC$^+$92] [GG99] [C93].

A consequence of this extended gcd computation is the first half of the following theorem [B96, 2.4] [L84, II.4, V.4]:

Theorem 2.2.3:
Every Euclidean ring is a principal ideal domain.
Every principal ideal domain is a unique factorization domain.

Being a *unique factorization domain* (UFD) means for $K[x]$ that any non-zero polynomial $f \in K[x]$ has a unique (up to order) factorization

$$f = \ell(f) \cdot \prod_{i=1}^{k} p_i^{e_i} \tag{2.3}$$

into its leading coefficient and powers of pairwise distinct monic *irreducible factors* p_i with their respective *multiplicities* $e_i > 0$. A polynomial $p \in K[x]$ is said to be *irreducible* if it is non-constant and has no *proper factors*, i. e. no non-constant factors of smaller degree. In any UFD R, the irreducible elements p are *prime*, meaning that $p|rs \Rightarrow p|r \vee p|s$ for all $r,s \in R$ [B96, 2.4] [L84, II.4].

Using unique factorization, it is easy to see that greatest common divisors are unique (up to units, which means up to non-zero constants for $K[x]$): Just regard taking the gcd of two ring elements as intersecting their multisets of irreducible factors. This allows us to define the symbol $\gcd(f,g)$ for two non-zero polynomials $f,g \in K[x]$ as their *unique* monic greatest common divisor.

Unique factorization in $K[x]$ has further implications:

Theorem 2.2.4: *(Gauss' Theorem)*
If R is a UFD, then $R[x]$ is also a UFD.

For a proof see [B96, 2.7] [L84, V.6]. It is easy to demonstrate the following consequence by induction:

Corollary 2.2.5: *If K is a field, then $K[x_1,\dots,x_n]$ is a UFD.*

Hence any non-zero $f \in K[x_1,\dots,x_n]$ factors uniquely into its irreducible factors with multiplicities as in (2.3). The existence of essentially unique gcds in multivariate polynomial rings is again a direct consequence of unique factorization, and we extend the notation $\gcd(f,g)$ introduced above to multivariate polynomials f and g.

Another simple consequence is this:

Corollary 2.2.6: *If R is a UFD, then $R[x_1,\dots,x_n]$ is a UFD.*

This allows us to speak of the essentially unique gcd of two non-zero polynomials $f,g \in R[x_1,\dots,x_n]$ and write $\gcd(f,g)$ in this case, too. This notion differs from the gcd in $\mathrm{Q}(R)[x_1,\dots,x_n]$, because non-unit constants are relevant for divisibility over R. For example, x divides $2x$ in $\mathbb{Z}[x]$, but $2x$ does not divide x.

One key step in the proof of Gauss' Theorem (see again [B96, 2.7] [L84, V.6]) clarifies exactly this relation of constant and polynomial factors and hence deserves to be mentioned here. When R is a UFD and $f \in R[x]$ a non-zero polynomial, the *content* $\mathrm{cont}(f)$ of f is the gcd of its coefficients. If $\mathrm{cont}(f) = 1$, then f is called *primitive*. The *primitive part* of f is the primitive polynomial $\mathrm{pp}(f) := f/\mathrm{cont}(f)$.

Proposition 2.2.7: *(Gauss' Lemma)*
Let R be a UFD, and let $f,g \in R[x]$ be non-zero polynomials. Then

$$\operatorname{cont}(fg) = \operatorname{cont}(f)\operatorname{cont}(g) \quad \textit{(up to units)}. \tag{2.4}$$

Corollary 2.2.8: *Let R be a UFD, and let $f,g \in R[x]$ be non-zero polynomials. Then*

$$\operatorname{cont}(\gcd(f,g)) = \gcd(\operatorname{cont}(f),\operatorname{cont}(g)) \quad \textit{(up to units)}. \tag{2.5}$$

Proof: The right-hand side $\gcd(\operatorname{cont}(f),\operatorname{cont}(g))$ is a constant factor of both f and g and consequently divides $\gcd(f,g)$ and its content. Conversely, $\operatorname{cont}(\gcd(f,g))$ is a constant factor of f and hence of $\operatorname{cont}(f)$, and the same holds for g, so that $\operatorname{cont}(\gcd(f,g))$ divides $\gcd(\operatorname{cont}(f),\operatorname{cont}(g))$. □

Corollary 2.2.9: *Let R be a UFD, and let $f,g \in R[x]$ be non-zero polynomials. Then the following conditions are equivalent:*

(i) $g|f$ *in* $R[x]$

(ii) $g|f$ *in* $\mathrm{Q}(R)[x]$ *and* $\operatorname{cont}(g)|\operatorname{cont}(f)$ *in* R.

In particular, these two notions of divisibility by g are equivalent if g is primitive.

Proof: The implication (i) $\Rightarrow$ (ii) is clear. To see (ii) $\Rightarrow$ (i), let $f = gh$ with $h \in \mathrm{Q}(R)[x]$. By multiplication with a common multiple of the denominators in the coefficients, we obtain $\tilde{h} = ch \in R[x]$ with $c \in R$. We can assume w. l. o. g. that $\operatorname{cont}(\tilde{h}) = 1$ (otherwise we would divide c by $\operatorname{cont}(\tilde{h})$).

It holds that $cf = g\tilde{h}$, which implies $c\operatorname{cont}(f) = \operatorname{cont}(g)\operatorname{cont}(\tilde{h}) = \operatorname{cont}(g)$ and $c^{-1} = \operatorname{cont}(f)/\operatorname{cont}(g) \in R$. Hence $h = c^{-1}\tilde{h} \in R[x]$. □

See Section A.1 for remarks on the essential role of unique factorization in this result.

These two corollaries have a rather useful consequence: When we can compute gcds in a UFD R, we can also compute $\gcd(f,g)$ for non-zero $f,g \in R[x]$: Because of Corollary 2.2.9, $\gcd(f,g)$ equals up to constants the gcd of f,g regarded as elements of $\mathrm{Q}(R)[x]$, and we can compute that with the Euclidean Algorithm. It is possible to avoid costly fractional arithmetic and perform this computation entirely within $R[x]$, yielding a constant multiple d of $\gcd(f,g)$. Because of Corollary 2.2.8, the error in the constant factor can be corrected by adjusting $\operatorname{cont}(d)$ to $c := \gcd(\operatorname{cont}(f),\operatorname{cont}(g))$, yielding $\gcd(f,g) = c \cdot \operatorname{pp}(d)$.

This extends to recursive algorithms for gcds in $K[x_1,\dots,x_n]$ or $R[x_1,\dots,x_n]$. We refer to [GC$^+$92] for details.

We conclude with the observation that gcd computation for $f,g \in K[x_1,\dots,x_n]$ does not depend on the coefficient field and is equally valid for all fields containing the coefficients of f and g.

2.2.3 Multiplicities and Derivatives

Whenever we have unique factorization of f into irreducibles, we can formulate the weaker concept of *square-free factorization,* or *factorization by multiplicities* as we like to call it. Take (2.3) and group factors p_i with equal multiplicities e_i into one factor:

$$f = \ell(f) \cdot \prod_{m=1}^{\max_i e_i} s_m^m, \qquad \text{where } s_m = \prod_{e_i=m} p_i \tag{2.6}$$

The factors s_m are *square free*, meaning that all *their* irreducible factors occur with multiplicity 1; and any two s_m are *coprime,* i. e. the two have no proper factor in common and thus a gcd of 1.

Using (2.6), we can define the *square-free part* of f as $\prod_m s_m$, i. e. we just reduce all exponents to 1.

The significance of this weaker form of polynomial factorization lies in the relative ease of computing it.

Let R be a ring and $f \in R[x]$ a polynomial. We can define the *(formal) derivative* of $f(x) = \sum_{i=0}^n a_i x^i$ as $f'(x) := \sum_{i=1}^n i a_i x^{i-1}$. (Our restricted definition of ring ensures that $a_i \neq 0 \;\Rightarrow\; i a_i \neq 0$.) This definition coincides with the rule from calculus for differentiating polynomial functions when $R = \mathbb{R}$ but does not involve analytic concepts (hence "formal"). Linearity of differentiation, the product rule and the chain rule remain valid. Differentiating r times is indicated by a parenthesized superscript $f^{(r)}$.

By taking a hierarchical view on multivariate polynomials and choosing different variables as outermost, the above definition extends to *partial derivatives,* indicated by subscripts such as $f_{x_1^3 x_2^2}$. Partial differentiation of polynomials w. r. t. different variables commutes. The result of differentiating $r \geq 0$ times w. r. t. some choice of variables is called an *r-th partial derivative* of f. The column vector of all first partial derivatives of f is the *gradient* $\nabla f := (f_{x_1}, \dots, f_{x_m})^{\mathsf{T}}$ of f.

The relation of derivatives and multiplicities is rooted in the following result.

Proposition 2.2.10: *Let K be a field, let $f \in K[x]$ be a non-zero polynomial, and let $f = \ell(f) \prod_{m=1}^M f_m^m$ be its square-free factorization. Then*

$$\gcd(f, f', \dots, f^{(k)}) = \prod_{m=k+1}^{M} f_m^{m-k} \tag{2.7}$$

for all $k \geq 0$. In particular, the square-free part of f is $f/\gcd(f, f')$.

Proof: By induction on k. The base case $k = 0$ is obvious. The inductive step rests on the following observation: Let $f = g^r h$, $r \geq 1$ so that g is irreducible and does not divide h. Then $f' = rg'g^{r-1}h + g^r h' = (rg'h + gh')g^{r-1}$, and g does not divide

$rg'h + gh'$, because it divides gh' but neither g' (which is of smaller degree) nor h (by definition). $\square$

This allows us to compute a square-free factorization of $f \in K[x]$ by repeated differentiation and gcd computation. Yun's Algorithm [GC$^+$92, 8.2] does this in a clever way to keep the coefficients of intermediate polynomials small. It is possible to extend this to multivariate polynomials. Proposition 2.2.1 implies that Proposition 2.2.10 remains valid up to constants when f is an x-regular polynomial from $K[x_1, \dots, x_{n-1}][x]$.

As before for gcds, note that square-free factorization does not depend on the coefficient field K and applies equally to all fields containing the coefficients of f.

2.2.4 Roots of polynomials

Let K be a field, and let $f \in K[x]$. Recall that as a consequence of division with remainder, $f(\xi) = 0$ at some point $\xi \in K$ iff $x - \xi$ is one of the irreducible factors of f. This allows us to define the *multiplicity* of ξ as a *root* or *zero* of f as the multiplicity of $x - \xi$ as a factor of f if $f(\xi) = 0$, and as 0 otherwise. Furthermore, it allows us to conclude that a polynomial of degree n has at most n roots, counted with multiplicity, and that an irreducible factor p of f is either linear, and hence corresponds to a zero of f, or has degree larger than 1 and no zeroes in K.

It is an important property of a field if the second alternative does not occur.

Theorem 2.2.11: *(Fundamental Theorem of Algebra)*
Let $f \in \mathbb{C}[x]$ be a polynomial of degree $n > 0$. Then f is a product of its leading coefficient and exactly n monic linear factors.

Fields like $\mathbb{C}$ with this property are called *algebraically closed*.

Theorem 2.2.12: *For every field K there exists an essentially unique $\subseteq$-smallest algebraically closed field $\overline{K}$ containing K, called the algebraic closure of K.*

For a proof see [B96, 3.4] [L84, VII.2].

The algebraic closure of $\mathbb{R}$ is $\mathbb{C} = \mathbb{R}(i)$, which can be considered to contain the algebraic closure $\overline{\mathbb{Q}}$ of $\mathbb{Q}$. This subset $\overline{\mathbb{Q}}$ is rather small, though, because it is countable (cf. §2.4.1), whereas $\mathbb{C}$, like $\mathbb{R}$, is uncountable.

On the algorithmic side, our main concern about roots is to determine the real roots of a polynomial $f \in \mathbb{Q}[x]$ in terms of isolating intervals with rational boundaries. An interval $[l, r] \subseteq \mathbb{R}$ is called an *isolating interval* for a root $\xi \in \mathbb{R}$ of $f \in \mathbb{R}[x]$ if $\{x \in [l, r] \mid f(x) = 0\} = \{\xi\}$.

A well-known algorithm to determine isolating intervals for all real roots of a square-free polynomial $f \in \mathbb{Q}[x]$ is *Uspensky's Method.*[1] It is based on the following classical Proposition,[2] attributed to René Descartes:

Proposition 2.2.13: *(Descartes' Rule of Signs)*
Let $f \in \mathbb{R}[x]$ be a non-zero polynomial with exactly p positive real roots (counted with multiplicity) and exactly v sign variations in its coefficient sequence. Then $v - p$ is non-negative and even.

A *sign variation* in a sequence $a_0, \ldots, a_n$ of real numbers is a pair of indices $0 \leq i < j \leq n$ such that $\operatorname{sign}(a_i)\operatorname{sign}(a_j) = -1$ and $a_{i+1} = \ldots = a_{j-1} = 0$. In other words, the number of sign variations is obtained by deleting all zeroes and counting how many pairs of adjacent numbers have opposite signs.

Proof: Let $f(x) = \sum_{i=0}^{n} a_i x^i$. Assume w. l. o. g. that $a_n > 0$, $a_0 \neq 0$. Since p is even iff $a_0 > 0$ and v is even iff $a_0 > 0$, their difference is always even. It remains to prove $v - p \geq 0$ by induction on $n = \deg(f)$. The base case $n = 0$ is clear.

For the inductive step, consider the derivative f'. Its number of sign variations is $v' \leq v$ and its number of positive real roots is $p' \geq p - 1$, since an $(m+1)$-fold root of f is an m-fold root of f', and there is a root of f' between any two adjacent roots of f. It follows that $v - p \geq v' - p' - 1 \geq -1$. Since $v - p$ is even, $v - p \geq 0$. $\square$

Proofs can also be found in [A89, Thm. 7.2.6] [M92, Thm. 5.5]. The following corollary is immediate:

Corollary 2.2.14: *In the situation of Proposition 2.2.13, the following implications hold:*

(i) If $v = 0$, then $p = 0$.

(ii) If $v = 1$, then $p = 1$.

The corollary extends to the number of real roots of f in an arbitrary open interval $]l, r[\neq \mathbb{R}$, $l, r \in \mathbb{R} \cup \{\pm\infty\}$ by considering the composition $f \circ T$ with a Möbius transformation $T(x) = \frac{ax+b}{cx+d}$ taking $]0, +\infty[$ to $]l, r[$, see [A89, 7.3.2]. This suggests to find isolating intervals for the real roots of a polynomial by starting from a bounded open interval containing all of them and subdividing it until each part is known to be an isolating interval or to contain no roots.

The crux is to prove termination, since the corollary does not contain equivalences, just implications. Termination can only hold for square-free polynomials f, since Descartes' Rule counts with multiplicities, and we cannot hope to isolate a multiple root against itself. Even for square-free f, the converse implications are false in cases where a complex root of f is "too close" to the positive real axis after

[1]This name is in common use but incorrect [RZ01] [A89, p. 396]. The credit should go to A. J. H. Vincent for his result of 1836 that justifies the basic algorithmic approach and not to J. V. Uspensky for his contribution of 1948.

[2]Its proof below has been adapted from lecture notes by Kurt Mehlhorn, unpublished, 2001.

transformation, but they hold once the interval under consideration is sufficiently small.

The first such partial converse is from Vincent (1836). We refer the reader to Akritas [A89] and Mignotte [M92] for more, but additionally point out a recently rediscovered theorem of Ostrowski [O50] with a particularly general sufficient condition for $v = 1$, see [KM0x]. See also [M01].

It remains to address the question of finding a first interval for the subdivision to start with. To get an upper bound B on the absolute value of all roots, determine

$$k_0 := \min\left\{ k \geq 0 \;\middle|\; |a_n|\, 2^{nk} > \sum_{i=0}^{n-1} |a_i|\, 2^{ik} \right\} \tag{2.8}$$

by successively trying $k = 0, 1, 2, 3, \ldots$ and taking $B = 2^{k_0}$, see [M92, p. 144].[3]

For the purposes of this text, it suffices to summarize that an efficient root isolation algorithm can be based on Descartes' Rule of Signs along the lines sketched above. See [A89, chap. 7] [RZ01].

An alternative to Uspensky's Method is provided by *Sturm Sequences*. These are polynomial remainder sequences of f and f' with special arrangements for alternating signs of the remainders [A89, 7.2.2] [M92, 5.4.1] [Y00, chap. 7]. Their values at interval boundaries indicate the exact number of real roots in the interval.

We prefer Uspensky's Method for our algorithm, on grounds of the working hypothesis that it is faster for square-free polynomials f, exploiting the fact that we can detect this quickly by checking $\det(\mathrm{Syl}(f, f')) \not\equiv 0$ modulo some prime (see §2.3.1). A clever implementation might nevertheless want to switch to Sturm Sequences for non-square-free polynomials, since computing their square-free part or their factorization by multiplicities already involves the computation of a remainder sequence for f and f'.

2.3 Elimination Theory

2.3.1 The Sylvester Resultant

Elimination Theory is a classical part of algebra concerned with the analysis and solution of systems of polynomial equations by reformulating algebraic properties of these systems as polynomial equations in their coefficients. The following basic proposition exemplifies this principle.

[3] The author has found this implemented in EXACUS, and a quick ad-hoc experiment indicated that it might perform better than the classical Cauchy Bound [M92, p. 192] [A89, Thm. 7.2.11].

Proposition 2.3.1: *Let K be a field. Let $f = \sum_{i=0}^{m} a_i x^i$, $a_m \neq 0$, and $g = \sum_{i=0}^{n} b_i x^i$, $b_n \neq 0$ be two polynomials in $K[x]$. Then the following conditions are equivalent:*

(i) *f and g have a common zero in the algebraic closure $\overline{K}$.*

(ii) $\deg(\gcd(f,g)) > 0$

(iii) *There are non-zero $u,v \in K[x]$ with $\deg(u) < \deg(g)$ and $\deg(v) < \deg(f)$ such that $uf + vg = 0$.*

(iv) *The determinant of the Sylvester matrix*

$$\operatorname{Syl}(f,g) = \begin{pmatrix} a_m & \cdots & \cdots & a_0 & & \\ & \ddots & & & \ddots & \\ & & a_m & \cdots & \cdots & a_0 \\ b_n & \cdots & \cdots & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_n & \cdots & \cdots & b_0 \end{pmatrix} \tag{2.9}$$

vanishes.

Proof: The equivalence of (i) and (ii) is clear in the light of §2.2.4.

For the implication (ii) $\Rightarrow$ (iii) let $d := \gcd(f,g)$, $u = g/d$, and $v = -f/d$. For the implication $\neg$(ii) $\Rightarrow$ $\neg$(iii) observe that whenever $uf + vg = 0$, then f divides vg but is coprime to g by $\neg$(ii) and hence divides v, such that $\deg(v) \geq \deg(f)$.

The equivalence of (iii) and (iv) follows by linear algebra, noting that the transpose $\operatorname{Syl}(f,g)^{\mathsf{T}}$ is the matrix of a linear map taking a pair of coefficient vectors from $K^n \times K^m$, representing degree-bound polynomials u and v, to the coefficient vector in K^{n+m} that represents the polynomial $uf + vg$. The determinant vanishes iff there is a non-zero vector that is mapped to zero. ☐

The determinant $\det(\operatorname{Syl}(f,g))$ is called the *(Sylvester) resultant* $\operatorname{res}(f,g,x)$ of f and g with respect to x.

The equivalence of (i) and (iv) can be summarized by stating: The resultant of two univariate polynomials f and g vanishes iff the overconstrained system $f = g = 0$ is solvable. The following proposition fits well into this picture:

Proposition 2.3.2: *Let K be a field. $f = a_m \prod_{i=1}^{m}(x - \alpha_i)$ and $g = b_n \prod_{j=1}^{n}(x - \beta_j)$ be two non-zero polynomials in $K[x]$. Then*

$$\operatorname{res}(f,g,x) = a_m^n b_n^m \prod_{i=1}^{m} \prod_{j=1}^{n} (\alpha_i - \beta_j) \tag{2.10}$$

A proof can be found in [B96, 4.4] [L84, V.10] [C93, 3.3.2] [W02, Thm. 2.5].

The Sylvester matrix has size $(m+n) \times (m+n)$. A more compact determinantal formulation for the resultant is the *Bezout matrix* of size $\max\{m,n\} \times \max\{m,n\}$, see [GC+92, 9.5].

2.3.2 Elimination with the Sylvester Resultant

Now consider the case of multivariate polynomials $f, g \in K[x_1, \dots, x_n]$. Proposition 2.3.1 can help us to analyze the set of solutions of the system $f = g = 0$ when we view f and g as polynomials in $\mathrm{Q}(K[x_1, \dots, x_{n-1}])[x_n]$. The analysis decomposes into two steps [CL$^+$97, 3.1]:

1. The *projection step* in which we determine the *partial solutions* $(\xi_1, \dots, \xi_{n-1})$ of $f = g = 0$, i. e. the values of $x_1, \dots, x_{n-1}$ for which the system $f = g = 0$ permits a solution $(\xi_1, \dots, \xi_n)$.

2. The *extension step* in which we extend a partial solution by all possible values ξ_n of x_n such that $f(\xi_1, \dots, \xi_n) = g(\xi_1, \dots, \xi_n) = 0$.

In this setting, $\mathrm{res}(f, g, x_n)$ is a polynomial from $K[x_1, \dots, x_{n-1}]$, and we might hope that its zero set is the set of partial solutions ξ, on the grounds that a vanishing resultant indicates solvability of $f|_\xi = g|_\xi = 0$. However, the definition of the resultant depends on the (univariate) degrees of f and g, and these degrees drop when ξ is a zero of $\ell(f)$ and $\ell(g)$. Then the Sylvester matrix $\mathrm{Syl}(f, g)|_\xi$ evaluated at ξ has a leftmost column full of zeroes, and so $\mathrm{res}(f, g, x_n)(\xi) = 0$ irrespective of ξ being a partial solution or not. Because of these extraneous zeroes, forming the resultant and evaluation in inner variables do not commute in general. Nevertheless, $\mathrm{res}(f, g, x_n)(\xi) = 0$ is a necessary condition for the existence of a solution extending ξ.

The situation is much better if one of the polynomials is x_n-regular.

Proposition 2.3.3: *Let K be a field, and let $f, g \in K[x_1, \dots, x_n]$ be non-zero polynomials. Furthermore, let f be x_n-regular. Then for all $(\xi_1, \dots, \xi_{n-1}) \in \overline{K}^{n-1}$ the two conditions*

(i) $\mathrm{res}(f, g, x_n)(\xi_1, \dots, \xi_{n-1}) = 0$

(ii) There is $\xi_n \in \overline{K}$ such that $f(\xi_1, \dots, \xi_n) = g(\xi_1, \dots, \xi_n) = 0$

are equivalent.

Proof: Let $\xi = (\xi_1, \dots, \xi_{n-1})$. We take a hierarchical view on the polynomials f and g with outermost variable x_n.

First we want to prove

$$\det(\mathrm{Syl}(f, g))|_\xi = 0 \iff \det(\mathrm{Syl}(f|_\xi, g|_\xi)) = 0. \tag{2.11}$$

Note that forming determinants does commute with evaluation. Recall that we have $\deg(f) = \deg(f|_\xi)$ and $\deg(g) \geq \deg(g|_\xi)$. Let $d := \deg(g) - \deg(g|_\xi)$. The matrix $\mathrm{Syl}(f, g)|_\xi$ contains $\mathrm{Syl}(f|_\xi, g|_\xi)$ as a lower right submatrix, plus d extraneous rows with coefficients of f at the top and d extraneous columns on the left which are empty below the diagonal. Therefore, we can expand d times by the respective first column and obtain

$$\det(\mathrm{Syl}(f,g)|_\xi) = \ell(f)^d \det(\mathrm{Syl}(f|_\xi, g|_\xi))$$

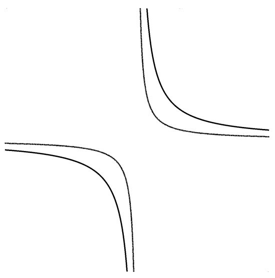
where $\ell(f)$ is a non-zero constant, proving (2.11).

Now we can conclude

$$\begin{aligned}
\text{(i)} &\iff \det(\mathrm{Syl}(f,g))|_\xi = 0 && \text{[by definition]}\\
&\iff \det(\mathrm{Syl}(f|_\xi, g|_\xi)) = 0 && \text{[by (2.11)]}\\
&\iff f|_\xi \text{ and } g|_\xi \text{ have a common root in } \overline{K} && \text{[by Proposition 2.3.1]}\\
&\iff \text{(ii)}
\end{aligned}$$

□

As an example demonstrating that the regularity of f is indeed necessary, consider $\mathrm{res}(xy-1,\, xy-3,\, y) = -2x$. You can visualize the system $xy-1 = xy-3 = 0$ as the intersection of two hyperbolae. Obviously, there is no solution extending $\xi_1 = 0$. It is possible to explain this failure using projective geometry by providing the "missing" solution at $x = 0$ as a point $(0:1:0)$ at infinity. We will come back to projective geometry in Section 3.1.



Using the resultant of f and g to find partial solutions algorithmically works better, of course, when the resultant is not the zero polynomial. The following result helps to attain this useful condition.

Proposition 2.3.4: *Let K be a field, and let $f,g \in K[x_1,\dots,x_n]$ be non-zero polynomials. Then* $\mathrm{res}(f,g,x_n) \neq 0$ *iff f and g have no common factor of positive degree in x_n.*

Proof: By Theorem 2.2.4, $K[x_1,\dots,x_{n-1}]$ is a UFD. Hence Corollary 2.2.9 implies that f and g have no common factor of positive univariate degree as elements of $K[x_1,\dots,x_{n-1}][x_n]$ iff they have no common factor of positive univariate degree as elements of $\mathrm{Q}(K[x_1,\dots,x_{n-1}])[x_n]$. By Proposition 2.3.1, this is equivalent to $\mathrm{res}(f,g,x_n) \neq 0$. □

Another practical concern relates to computations over the field of real numbers $\mathbb{R}$ which is not algebraically closed. The following proposition helps to recognize cases in which a partial solution $\xi \in \mathbb{R}^{n-1}$ extends to a solution in $\mathbb{R}^n$ (as opposed to $\mathbb{C}^n$). Recall that complex conjugation $\bar{\cdot} : \mathbb{C} \to \mathbb{C}$, $a+ib \mapsto a-ib$ is a ring automorphism of $\mathbb{C}$ that keeps precisely the elements of $\mathbb{R}$ fixed.

Proposition 2.3.5: *Let $f_1 = f_2 = \ldots = f_k = 0$ be a polynomial system of equations with $f_1,\dots,f_k \in \mathbb{R}[x_1,\dots,x_n]$ which has exactly $m > 0$ solutions $\xi_1,\dots,\xi_m \in \mathbb{C}^n$, exactly r of which are real. Then the remaining $m-r$ solutions form complex-conjugate pairs. In particular, $m-r$ is even, and $r \geq 1$ whenever m is odd.*

Proof: Let $\xi_i = (\xi_{i1}, \ldots, \xi_{in}) \in \mathbb{C}^n \setminus \mathbb{R}^n$ be one of the non-real solutions. Then its complex conjugate $\overline{\xi_i} = (\overline{\xi_{i1}}, \ldots, \overline{\xi_{in}})$ is another non-real solution, because the real coefficients of all f_j are fixed under complex conjugation, such that $f_j(\overline{\xi_i}) = \overline{f_j(\xi_i)} = 0$ for all j. Since $\overline{\overline{\xi_i}} = \xi_i$, the complex-conjugate solutions come in pairs, totalling an even number of $m - r$ of non-real solutions. $\square$

2.3.3 Subresultants

Proposition 2.3.1 can be read as a statement on the polynomial remainder sequence of f and g: The resultant vanishes iff the remainder sequence terminates without a non-zero constant occurring as the gcd. This relation can be generalized, using the following notion:

Let K be a field, and let $f, g \in K[x]$ be two non-zero polynomials with degrees no less than $k \geq 0$. Then the k-th *(scalar) subresultant* $\mathrm{sres}_k(f,g,x)$ of f and g is defined as the determinant of the matrix obtained from $\mathrm{Syl}(f,g)$ by deleting the last $2k$ columns of the matrix and the last k rows of each of the two parallelograms of coefficients. (A recent research article [HW00] contains a more compact formulation of the subresultant with a generalized Bezout matrix.)

Proposition 2.3.6: *With f, g and k as above, the remainder sequence computed by the Euclidean algorithm for f and g contains a non-zero remainder of degree k iff the k-th subresultant of f and g does not vanish.*

Textbook references for a proof are [GG99, 6.10] [Kn97, 4.6.1] [GC$^+$92, 7.3].

This result deserves to be mentioned because it hints at a deep connection between polynomial remainder sequences and subresultants: Non-vanishing subresultants are factors of the polynomial remainders with the corresponding degree, and one can compute them incrementally for decreasing values of k together with the remainder sequence. Dividing the polynomial remainders by these known constant factors provides an effective means for the *subresultant algorithm* by Collins (1967) and Brown and Traub (1971) to reduce their coefficients lengths and to attain a polynomially bounded running time.

The subresultant algorithm is described in [C93] [GC$^+$92] [Kn97]. The relation of subresultants and polynomial remainder sequences is also covered in [GG99] (with historical notes on p. 188) and in the useful survey [L00]. The latter also clarifies the relation of the conflicting definitions of scalar and polynomial subresultant found in the literature and contains an empirical comparison of the subresultant algorithm with the competing idea of dividing every polynomial remainder by its content.

For the rest of our investigations, we will only employ the following immediate corollary. It is stated explicitly in [GC$^+$92, Thm. 7.3]. A direct proof is in Wolpert [W02, Prop. 2.11].

Corollary 2.3.7: *With f, g and k as above,*

$$\mathrm{sres}_i(f,g,x) = 0 \textit{ for all } 0 \le i < k \iff \deg(\gcd(f,g)) \ge k. \tag{2.12}$$

2.4 Algebraic Numbers

2.4.1 Algebraic Number Fields

Let $\vartheta \in \mathbb{C}$ be a zero of a non-zero polynomial $f \in \mathbb{Q}[x]$. Since $\mathbb{Q}[x]$ is countable and every non-zero polynomial has only finitely many complex roots, there are only countably many such complex numbers ϑ. We call these special numbers *algebraic numbers*. Mathematics has a rich theory on algebraic numbers, see [B96] [L84] [C93].

From §2.2.2 we know that we can always choose f to be irreducible and monic. This determines f uniquely and we call it the *minimal polynomial* of ϑ. Clearly, $f|h$ whenever $h(\vartheta) = 0$ for some $h \in \mathbb{Q}[x]$. The degree $n := \deg(f)$ is called the *degree* $\deg(\vartheta)$ of ϑ and measures the irrationality of ϑ: It is rational iff $n = 1$. Otherwise the $\subseteq$-smallest field containing $\mathbb{Q}$ and ϑ, denoted $\mathbb{Q}(\vartheta)$, is strictly larger than $\mathbb{Q}$, and the following proposition quantifies this in terms of n.

Proposition 2.4.1: *Let ϑ be an algebraic number of degree $n > 1$ with minimal polynomial f. Then*

$$\begin{aligned} \iota : \mathbb{Q}[x]/(f) &\to \mathbb{Q}(\vartheta) \\ \overline{x} &\mapsto \vartheta \end{aligned} \tag{2.13}$$

is an isomorphism of rings. Furthermore, $\mathbb{Q}[x]/(f)$ is an n-dimensional $\mathbb{Q}$-vector space with basis

$$\{\overline{1}, \overline{x}, \overline{x^2}, \ldots, \overline{x^{n-1}}\}. \tag{2.14}$$

The slash in $\mathbb{Q}[x]/(f)$ denotes a *quotient ring*, similar to $\mathbb{Z}/(p)$ (the integers modulo a prime p). Hence the elements of $\mathbb{Q}[x]/(f)$ are equivalence classes of $\mathbb{Q}[x]$ under the congruence relation $g \equiv_f h \;:\Leftrightarrow\; f|(g-h)$, and the operations defined on representatives make this a ring again. We write $\overline{h}$ for the equivalence class $[h]_{\equiv_f}$ of h.

Proof: The evaluation homomorphism $\mathbb{Q}[x] \to \mathbb{Q}(\vartheta)$, $x \mapsto \vartheta$ maps exactly the multiples of f to 0 and hence maps polynomials g and h to the same element of $\mathbb{Q}(\vartheta)$ iff $g \equiv_f h$. Therefore ι is well-defined and injective.

$\mathbb{Q}[x]/(f)$ is a field, because a non-zero element $\overline{h} \in \mathbb{Q}[x]/(f)$ is represented by a polynomial h coprime to the irreducible polynomial f, so that the Extended Euclidean Algorithm produces Bezout factors u, v with $1 = \gcd(f,h) = uf + vh$, implying that $\overline{v}\overline{h} = \overline{1}$.

Thus, the image of the injective ring homomorphism ι is a field, too. Since it contains ϑ, it contains $\mathbb{Q}(\vartheta)$, meaning that ι is surjective and hence an isomorphism.

Every $\overline{h} \in \mathbb{Q}[x]/(f)$ has a unique representative of degree less than n, viz. the remainder of h under Euclidean division by f. This is the unique representation of h as a $\mathbb{Q}$-linear combination of $\{\overline{1}, \overline{x}, \overline{x^2}, \ldots, \overline{x^{n-1}}\}$. Therefore, this set is a $\mathbb{Q}$-vector space basis of $\mathbb{Q}[x]/(f)$. $\square$

A field like $\mathbb{Q}(\vartheta)$ that is a superfield of $\mathbb{Q}$ and has a finite $\mathbb{Q}$-vector space dimension is called an *(algebraic) number field*. All number fields, or equivalently all fields obtained by adjoining a finite number of algebraic numbers to $\mathbb{Q}$, can be generated by adjoining just one *primitive element* ϑ, see [B96, 3.6] [L84, VII.6]. The dimension of a number field K as a $\mathbb{Q}$-vector space is an intrinsic property of the field which we call the *degree* $\deg(K)$ of the field. For all elements $\alpha \in K = \mathbb{Q}(\vartheta)$ we have $\deg(\alpha) \leq \deg(\vartheta) = \deg(K)$.

An interesting immediate consequence of the isomorphism in Proposition 2.4.1 is the idea of conjugacy of algebraic numbers, which is in perfect analogy to complex conjugation in $\mathbb{C} = \mathbb{R}(i)$.

Corollary 2.4.2: *Let ϑ_1 be an algebraic number of degree $n > 1$ with minimal polynomial f. Then there are exactly n complex roots $\vartheta_1, \ldots, \vartheta_n$ of f, called the conjugates of ϑ_1; and $\vartheta_i \mapsto \vartheta_j$ induces an isomorphism $\mathbb{Q}(\vartheta_i) \simeq \mathbb{Q}(\vartheta_j)$ for all $1 \leq i, j \leq n$ which keeps the elements of $\mathbb{Q}$ fixed.*

This means that a computation over the field $\mathbb{Q}(\vartheta)$ that involves only field operations and comparisons for (in)equality produces the same result for all conjugates of ϑ. In particular, we can compute gcds, derivatives, resultants, etc. in $\mathbb{Q}(\vartheta)[x_1, \ldots, x_n]$ irrespective of the choice of a conjugate ϑ. We have the following analogue to Proposition 2.3.5.[4]

Proposition 2.4.3: *Let $f_1 = f_2 = \ldots = f_k = 0$ be a system S of polynomial equations with $f_1, \ldots, f_k \in \mathbb{Q}[x_1, \ldots, x_n]$ which has a solution $\xi \in \mathbb{Q}(\vartheta)^n$ for some algebraic number $\vartheta \in \mathbb{C}$.*

Let ϑ' be any conjugate of ϑ, and let $\iota : \mathbb{Q}(\vartheta) \to \mathbb{Q}(\vartheta')$ be the isomorphism induced by $\vartheta \mapsto \vartheta'$. Then $\iota(\xi) \in \mathbb{C}^n$ also is a solution of S.

If ϑ is chosen to have the smallest possible degree, then each choice of a conjugate ϑ' of ϑ yields a conjugate $\iota(\xi)$ of ξ which is different from all others. Hence irrational solutions $\xi \in \mathbb{C}^n$ to S come in groups of $\deg(\vartheta)$ *conjugates.*

Proof: The proof of the first statement is the same as for Proposition 2.3.5, with ι in place of complex conjugation.

[4]The proof is similar to a proof given by Susanne Schmitt in personal communication to Kurt Mehlhorn, March 2003.

To see the second statement, observe that choosing ϑ to minimize $\deg(\vartheta)$ under the constraint $\xi \in \mathbb{Q}(\vartheta)^n$ means choosing a primitive element of $\mathbb{Q}(\xi_1, \ldots, \xi_n) = \mathbb{Q}(\vartheta)$. Hence ϑ can be written as a rational function $\vartheta = P(\xi)/Q(\xi)$ with polynomials $P, Q \in \mathbb{Q}[x_1, \ldots, x_n]$. Now assume two isomorphisms ι and ι' of the kind in question agree on ξ. The coefficients of P and Q are fixed under ι and ι', hence $\iota(\vartheta) = \iota(\vartheta')$ and thus $\iota = \iota'$. □

The proof motivates us to define the *degree* $\deg(\xi)$ of an n-tuple ξ of algebraic numbers as the degree of the number field $\mathbb{Q}(\xi_1, \ldots, \xi_n)$ generated by its elements. The conjugates of ξ can be defined without reference to ϑ as the $\deg(\xi)$ different images of ξ under the $\deg(\xi)$ different embeddings of $\mathbb{Q}(\xi_1, \ldots, \xi_n)$ into $\mathbb{C}$.

The nice symmetric behaviour of algebraic conjugates under field operations and (in)equality comparisons changes drastically as soon as three-valued comparisons testing for less-equal-greater come into play. Consider the example $f = x^4 - 2$. This irreducible polynomial has four roots, two of which are not even real and thus do not allow meaningful three-valued comparisons, and the other two roots are real, but behave differently under comparison with 0, say, because one is positive and the other is negative.

We conclude that the algebraic and analytic properties of algebraic numbers are separate matters. Indeed, we will see ways of computing with them that address just one or the other of these two aspects, and we will see useful applications of the symmetry resulting from algebraic conjugacy.

2.4.2 Arithmetic for Algebraic Numbers

Proposition 2.4.1 and its proof immediately suggest a way of computing in number fields: Compute with the canonical representatives of $\mathbb{Q}[x]/(f)$. The vector space operations are straightforward. Multiplication is possible by multiplication of representatives followed by taking the remainder modulo f. As an optimization, one can precompute a linear-size multiplication table for products of basis elements $\overline{1}, \ldots, \overline{x^{n-1}}$. Division requires an extended gcd computation. Since we work with canonical representatives, comparison for (in)equality is straightforward.

A major drawback of this approach is that it just works within a fixed number field. Operations that mix two numbers from different number fields of degrees m and n lead into a larger number field with a degree $\leq mn$ and require the synthesis of a new minimal polynomial.

More on this and alternative approaches can be found in [C93, 4.2].

It is possible to extend this approach to three-valued comparisons of algebraic numbers in $\mathbb{Q}(\vartheta) \subseteq \mathbb{R}$ when we can approximate the real zero ϑ of f sufficiently well. To compare two numbers $\alpha, \beta \in \mathbb{Q}(\vartheta)$, compute the canonical representative h of $\alpha - \beta$ in $\mathbb{Q}[x]/(f)$ which is a rational polynomial of degree $\deg(h) \leq n - 1$. If

$\alpha = \beta$, then $h = 0$, and we are done. Otherwise, standard methods of numerical analysis allow us to determine a sufficiently accurate approximation $\tilde{\vartheta}$ of ϑ such that $\operatorname{sign}(h(\tilde{\vartheta})) = \operatorname{sign}(h(\vartheta)) = \operatorname{sign}(\alpha - \beta) = \pm 1$.

So far we have taken for granted that we know an irreducible polynomial $f \in \mathbb{Q}[x]$ defining an algebraic number ϑ. In practice, we will often have to start with just *any* polynomial having ϑ as a root. Using factorization by multiplicities (2.6), we can easily replace f by an appropriate square-free factor, but factoring into irreducibles is much harder. So let us see what happens if we are bold and just work with a monic square-free modulus f which factors into $r > 1$ irreducibles $f = \prod_{i=1}^{r} p_i$.

The central problem is that $\mathbb{Q}[x]/(f)$ contains zero divisors. For example, we have $\overline{f/p_1} \cdot \overline{p_1} = \overline{0}$, although both factors are non-zero. Hence for the rest of this section, we drop the restriction that the rings we consider be free of zero divisors.

Proposition 2.4.4: *(Chinese Remainder Theorem for polynomials)*
Let $f = \prod_{i=1}^{r} p_i \in \mathbb{Q}[x]$ *be a product of* $r > 1$ *pairwise coprime irreducibles. Then*

$$\begin{array}{rcl} \iota : \mathbb{Q}[x]/(f) & \to & \mathbb{Q}[x]/(p_1) \times \cdots \times \mathbb{Q}[x]/(p_r) \\ \overline{h} & \mapsto & (\overline{h}, \ldots, \overline{h}) \end{array} \tag{2.15}$$

is an isomorphism of rings.

The cartesian product $R \times S$ of two rings R and S is again a ring. Its elements are the pairs (r, s), and operations on them are defined component-wise. This extends to more than two factors in a natural way. In the formula above, the bars on the right-hand side denote equivalence classes $[h]_\equiv$ under congruence w. r. t. the various p_i.

Proof: The map ι is well-defined, because the moduli p_i are factors of f. It is obviously a homomorphism of rings. It is injective, because $\iota(\overline{g}) = \iota(\overline{h})$ implies that all p_i divide $g - h$, so that f divides $g - h$ as well, which means $\overline{g} = \overline{h}$.

To demonstrate surjectivity, let $(\overline{g}_1, \ldots, \overline{g}_r) \in \mathbb{Q}[x]/(p_1) \times \cdots \times \mathbb{Q}[x]/(p_r)$ be any element. Let $h = \sum_{i=1}^{r} g_i e_i (f/p_i)$ where each $e_i \in \mathbb{Q}[x]$ represents $(\overline{f/p_i})^{-1}$ modulo p_i. Then $\iota(\overline{h}) = (\overline{g}_1, \ldots, \overline{g}_r)$. $\square$

This elucidates why $\mathbb{Q}[x]/(f)$ contains zero divisors $\overline{h}$: The product of two non-zero r-tuples is zero iff for each component there is a zero in one of the factors. We can also interpret the proposition in terms of the algebraic number ϑ we are trying to compute with: An expression in a zero ϑ of f may or may not evaluate to 0 depending on the irreducible factor p_i of f that vanishes at ϑ.

We can overcome this algorithmically as follows: Given a root ϑ of a square-free polynomial $f \in \mathbb{Q}[x]$, compute with canonical representatives in $\mathbb{Q}[x]/(f)$ as usual. The ring operations addition, subtraction, and multiplication work seamlessly. Division by $\overline{h}$ requires an extended gcd computation. If $d := \gcd(f, h) = 1$, then $\overline{h}$ is invertible modulo all p_i and we can continue as before. Otherwise, we have found a proper factor d of f. If d vanishes at ϑ, then h represents 0 and is not invertible; else we resume computation with the new modulus $f/d \in \mathbb{Q}[x]$ that vanishes at ϑ and is coprime to h. Comparison for (in)equality is similar: Two canonical representatives g,h are equal iff they represent the same number modulo all p_i, but this is not what we need for computation with a specific ϑ. Instead, compute $d := \gcd(f, g-h)$. The two resulting factors d and f/d of f comprise exactly those p_i modulo which equality does or does not hold, respectively, and we can decide for the one that vanishes at ϑ.

Sometimes one can turn the argument around: If we know for some reason that all expressions in ϑ which we compute are simultaneously zero or non-zero for all choices of a root ϑ of f, then we can drop the precautions developed above and compute in $\mathbb{Q}[x]/(f)$ as if f was irreducible.

Finally, there is a way to ensure irreducibility of the modulus $f \in \mathbb{Q}[x]$ in a simple way if $\deg(f) \leq 3$. Then f is either irreducible or contains a linear factor g, corresponding to a rational zero. Hence factoring a polynomial of degree ≤ 3 into irreducibles amounts to finding rational zeroes. We state results that indicate how to do this.

Proposition 2.4.5: *Let $f \in \mathbb{Z}[x]$ be a polynomial, and let $a,b \in \mathbb{Z}$ such that $f(\frac{a}{b}) = 0$, $\gcd(a,b) = 1$. Then $b | \ell(f)$. In particular, all rational zeroes of a monic integer polynomial f are integers.*

Proof: The polynomial $g = bx - a$ divides f in $\mathbb{Q}[x]$ and is primitive. Then Corollary 2.2.9 implies $g|f$ in $\mathbb{Z}[x]$ so that $b = \ell(g) | \ell(f)$ in $\mathbb{Z}$. $\square$

To apply the particular case of this corollary, consider $\hat{f}(x) = \ell(f)^{\deg(f)-1} f(\frac{x}{\ell(f)})$. Observe that $\hat{f} \in \mathbb{Z}[x]$ is monic. Standard methods of numerical analysis allow to obtain approximations $\tilde{\xi}$ of all real roots ξ of $\hat{f}$ with an error < 0.5. Round every $\tilde{\xi}$ to the nearest integer $[\tilde{\xi}]$ and check whether $\hat{f}([\tilde{\xi}]) = 0$. If yes, $\xi = [\tilde{\xi}]$ is integer and corresponds to a rational root $\frac{\xi}{\ell(f)}$ of f. If no, ξ is irrational and corresponds to an irrational root of f. (For $\deg(f) = 2$, this method amounts to checking whether the discriminant has an integer square root.)

2.4.3 Arithmetic based on Separation Bounds

A frequent characteristic of geometric applications of real algebraic numbers is that the expressions computed with them are often positive or negative with a large magnitude and close to zero only in degenerate or almost-degenerate situations. It is therefore beneficial if the numerically easy cases are cheap to compute. Representing algebraic numbers in $\mathbb{Q}[x]/(f)$ does not have this property at all: Any operation has to be performed symbolically, involving long coefficients.

A completely different approach has therefore been taken by the developers of the number types `leda::real` [BFMS00] [MN99, 4.4] and `CORE::expr` [KL$^+$99]. For these number types, arithmetic operations do not actually do arithmetic, they just build up expression trees. Only when a sign is to be determined (for example to compare two numbers), the expression tree is evaluated numerically at increasing levels of precision. If the sign is non-zero, the precision will eventually suffice to recognize this with certainty. The crucial point, however, is how to recognize the sign zero. This is facilitated by a *separation bound* (or *constructive root bound*) B, an efficiently computable attribute of an expression E such that $E \neq 0 \Rightarrow |E| \geq B$. Hence if numerical approximation indicates that E lies in $]-B, B[$, it is certainly 0. Simple cases where $|E|$ is much larger than B require only a rough approximation. Detecting $E = 0$ is expensive, though.

The ranges of numbers represented by `leda::real` and `CORE::expr` contain $\mathbb{Q}$ and are closed under field operations and certain root extraction operations. More specifically, `CORE::expr` currently allows real square roots (thus modelling the field of all lengths one can construct with straightedge and compass from a unit length), whereas `leda::real` allows real k-th roots for arbitrary integer $k \geq 2$ and is, at the time of this writing, being extended by Susanne Schmitt to allow real zeros of arbitrary square-free polynomials (denoted by the so-called *diamond operator* [BFMSS01]).

The ability to freely nest these operations on `leda::real`s and `CORE::expr`s, i. e. to extract roots from radicands/polynomials that are themselves root expressions, is impressive but comes at the price of deteriorating separation bounds. On the opposite end, the floating point filtering inherent in these number types even seems to be helpful in some cases when roots are not used at all to speed up computations involving long rationals [Hert02].

This separation bound approach to computing with algebraic numbers is best suited to cases where some algebraic numbers are constructed once and then the signs of various expressions involving them are determined, with zero or almost-zero occurring seldomly. This excludes certain algebraic operations, for example computing polynomial remainder sequences with the Euclidean Algorithm: Each division with remainder intermixes the coefficients, resulting in quickly growing expression trees, and zeroes are produced on purpose. Here representations like $\mathbb{Q}[x]/(f)$ benefit from their property that representations simplify automatically to a canonical form.

For expressions involving one square root $\sqrt{D}$ it is easy to implement a number type for computing symbolically in $\mathbb{Q}(\sqrt{D})$ with the "pencil and paper" method. (In fact, Michael Hemmer has contributed such a number type to the implementation.) This is essentially equivalent to computing in $\mathbb{Q}[x]/(x^2 - D)$.

A combination of symbolic and separation bound arithmetic in the following style will prove helpful later on: Perform algebraic operations over $\mathbb{Q}(\sqrt{D})$ with the symbolic representation, convert to a separation bound number type, and continue with operations like nested root extraction and comparisons benefitting from the separation bound approach.

2.4.4 Real Algebraic Numbers as Isolated Roots

Extraction of k-th real roots alone does not suffice to solve general polynomial equations of degree larger than 2. (This may seem to contradict the classical algebraic theorem whereby the degree limit for solvability by radicals is 4. However, "radicals" in that sense are arbitrary, not just real, zeroes of polynomials $x^k - c$ [L84, VIII.7] [B96, 6.1].) Confronted with this situation while working on [BE$^+$02], Michael Hemmer and Kurt Mehlhorn developed a method that is closer to §2.4.2, simpler to implement, and not necessarily inferior to §2.4.3 if just comparisons of roots and no arithmetic operations are required:

A real algebraic number ϑ is represented by a pair $(f,[l,r])$ consisting of a squarefree polynomial $f \in \mathbb{Q}[x]$ vanishing at ϑ and an isolating interval $[l,r] \ni \vartheta$ with boundaries $l,r \in \mathbb{Q}$. Except in the degenerate case $l = r$, we demand $\vartheta \in]l,r[$. (A reversed bracket indicates exclusion of the boundary point.)

Isolating intervals for all real roots of a rational polynomial can be determined, for example, using Uspensky's Algorithm or Sturm Sequences, see §2.2.4.

A key step in the comparison of such algebraic numbers is the bisection of a nondegenerate isolating interval $[l,r]$ at a point $t \in]l,r[$. By Proposition 2.2.10, the squarefreeness of f implies that $f'(\vartheta) \neq 0$, hence the continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ does not have a local extremum at ϑ and consequently changes sign at ϑ. The absence of further zeroes in $[l,r]$ together with the Intermediate Value Theorem implies that $\pm 1 = \operatorname{sign}(f(l)) \neq \operatorname{sign}(f(r)) = \mp 1$. By comparing $\operatorname{sign}(f(t))$ against $\operatorname{sign}(f(r))$, 0, and $\operatorname{sign}(f(l))$, we can decide whether $\vartheta \in]l,t[$, $\vartheta = t$, or $\vartheta \in]t,r[$, and refine $[l,r]$ to $[l,t]$, $[t,t]$ or $[t,r]$, respectively.

Let $\vartheta_1 = (f_1,[l_1,r_1])$ and $\vartheta_2 = (f_2,[l_2,r_2])$ be two algebraic numbers that we intend to compare. If their isolating intervals overlap, we bisect each interval at the boundaries of the other interval it contains to make the two intervals non-overlapping or equal. The order of two algebraic numbers with non-overlapping isolating intervals is apparent from their interval boundaries.

So let us assume that by now $[l_1,r_1] = [l_2,r_2] := [l,r]$ with $l < r$. The polynomials f_1 and f_2 are square free and each has exactly one zero in $[l,r]$. Hence $d := \gcd(f,g)$ is also square free and has at most one zero in $[l,r]$. We have

$$\begin{aligned}
\operatorname{sign}(d(l)) \neq \operatorname{sign}(d(r)) &\iff \text{There is } \eta \in [l,r] \text{ s.t. } \quad d(\eta) = 0 \\
&\iff \text{There is } \eta \in [l,r] \text{ s.t. } \quad f_1(\eta) = 0 \wedge f_2(\eta) = 0 \\
&\iff \vartheta_1 = \vartheta_2.
\end{aligned}$$

If $\operatorname{sign}(d(l)) \neq \operatorname{sign}(d(r))$ holds, we are done. Otherwise, $\vartheta_1 \neq \vartheta_2$, and we alternately refine their isolating intervals $[l_i, r_i]$ by bisection at the midpoint $\frac{1}{2}(l_i + r_i)$ until they are non-overlapping. Termination is guaranteed by the fact that the interval lengths converge to 0 so that their sum has to drop below $|\vartheta_1 - \vartheta_2|$ eventually.

When we have computed a non-constant gcd d, we can replace either f_i by d or f_i/d, whichever vanishes at ϑ_i. This pays off in future comparisons because it lowers the degree of f_i, making evaluation and gcd computation cheaper.

Two optimizations are possible to avoid gcd computations. The first is to refine the intervals a number of times and hope to determine inequality in this way before computing the gcd to check for equality. The second is to evaluate the determinant of the Sylvester or Bezout matrix of f_1 and f_2 modulo a prime number p. If the result is non-zero, then $\operatorname{res}(f_1, f_2, x) \neq 0$ and hence $\gcd(f_1, f_2) = 1$. The converse, of course, does not hold. This is much cheaper than computing the true resultant or gcd, because it avoids costly arithmetic with long numbers.

Another desirable optimization is to refine intervals by a more efficient method than bisection at midpoint. Bisection essentially produces one more bit of ϑ per iteration. Once the intervals are small enough, one could do Newton-Raphson iteration instead, which would double the number of known bits of ϑ in each iteration. (Work in this direction is being done for the diamond operator.)

This representation of real algebraic numbers can be seen as a delayed form of numerical root finding (albeit with a portion of symbolic computation to handle equality accurately). Later comparisons benefit from the refinements done earlier. Past inequality results are cached in the form of non-overlapping isolating intervals. Additionally, one can implement a union-find scheme to cache equality, replacing the isolating intervals $[l_1, r_1], [l_2, r_2]$ of two equal numbers being united by their intersection $[l_1, r_1] \cap [l_2, r_2]$. Firstly, this causes all logical consequences of transitivity, reflexivity and symmetry of $=$ to be evident from unitedness. Secondly, all logical consequences of transitivity of $>$ previously reflected by $[l_1, r_1]$ or $[l_2, r_2]$ are also reflected by their intersection. Therefore, any logical consequence of a sequence of three-valued comparisons of algebraic numbers is already reflected by their interval boundaries.

Finally, a further chance for optimization is to use `leda::reals` instead of isolating intervals for the comparison of algebraic numbers of degree 2, since they can always be expressed using real square roots.

We will need one more operation on algebraic numbers, viz. refining the isolating interval $[l, r]$, $l \neq r$, of a zero ϑ of f against those zeroes of another square-free polynomial $g \in \mathbb{Q}[x]$ different from ϑ. To achieve this, refine $[l, r]$ by bisection until one of the following conditions holds:

- Descartes' Rule of Signs implies the absence of a zero of g in the interval.

- Descartes' Rule of Signs implies the existence of at most one zero of g in the interval; and a sign change of $\gcd(f,g)$ at the boundaries implies the existence of at least one common zero. (This zero must then be unique and identical to ϑ, since we consider isolating intervals of ϑ with respect to f.)

For this specific operation, we want to retain an interval containing ϑ in its interior, so the special case of hitting a zero exactly with the bisection point needs to be resolved by choosing another bisection point.

The modular coprimality check mentioned above can also be employed here to avoid most computations of constant gcds.

Chapter 3

Geometry of Algebraic Curves

This chapter is dedicated to the study of algebraic curves, and especially those of degree up to 3. The first section covers projective geometry, since $\mathbb{R}^2$ or even $\mathbb{C}^2$ are too restrictive to exhibit all features of the algebraic curves within them. The second section studies algebraic curves of arbitrary degree and aspects of their real geometry, excluding singularities.

It is only in the third section that the restriction to degree ≤ 3 is actually made, allowing a complete investigation of all singularities involved. Additionally, the algebraic degrees of special points are investigated. The chapter concludes with a section on parametrized curves, bringing cubic splines into the scope of our algorithm.

The main reference used is Gibson's very accessible introduction [G98]. Additional references are Cox-Little-O'Shea [CL$^+$97] and the advanced book by Brieskorn and Knörrer [BK86]. Another introductory text is Bix [B98].

Some basic analytic and topological concepts are used to make the link between algebra and geometry, and the reader is assumed to be familiar with them, at least on an intuitive level.

3.1 Projective Geometry

3.1.1 Projective Spaces

Let K be a field and $n > 0$ an integer. Consider the $(n+1)$-dimensional *affine space* K^{n+1}, which is also a K-vector space, and the following equivalence relation on it. Two points $v, w \in K^{n+1}$ are *collinear*, denoted $v \| w$, iff there is a non-zero $\lambda \in K$ such that $v = \lambda w$. The origin $0 = (0, \ldots, 0)$ forms an equivalence class $\{0\}$ of its own. The other equivalence classes under $\|$ are punctured lines through the origin:

$$(v_1 : v_2 : \ldots : v_{n+1}) := [(v_1, v_2, \ldots, v_{n+1})]_{\|} = \{\lambda v \mid 0 \neq \lambda \in K\} \tag{3.1}$$

for non-zero $v \in K^{n+1}$. The n-dimensional *projective space* over K is defined as the set of these punctured lines:

$$\mathbb{P}^n(K) := (K^{n+1} \setminus \{0\}) \,/\, \|. \tag{3.2}$$

We can regard the n-dimensional affine space as a subset of the n-dimensional projective space over K by embedding it as follows:

$$\begin{array}{rrcl} \iota : & K^n & \hookrightarrow & \mathbb{P}^n(K) \\ & (v_1, \ldots, v_n) & \mapsto & (v_1 : \ldots : v_n : 1) \end{array} \tag{3.3}$$

In the affine model K^{n+1} of $\mathbb{P}^n(K)$, the set of points $(v_1, \ldots, v_n, 1)$ is an n-dimensional hyperplane which contains exactly one representative for every projective point $(v_1 : \ldots : v_n : 1)$ but none for the remaining projective points $(w_1 : \ldots : w_n : 0)$.

For the moment, let K be $\mathbb{R}$ to provide us with a notion of convergence. Observe that $(cw_1 + v_1 : \ldots : cw_n + v_n : 1) = (w_1 + \frac{v_1}{c} : \ldots : w_n + \frac{v_n}{c} : \frac{1}{c})$, so that shooting a ray $(cw_1 + v_1, \ldots, cw_1 + v_n)$, $c \to \infty$ in the affine space makes the corresponding points in the projective space converge to a point $(w_1 : \ldots : w_n : 0)$. In this sense, the points $(\ldots : 1)$ are finite points of affine space and the points $(\ldots : 0)$ are points at infinity. Observe that the infinite point $(w_1 : \ldots : w_n : 0)$ depends only on the (unoriented) direction $\pm(w_1, \ldots, w_n)$ of the ray, so that parallel lines have one common point at infinity.

Actually, the points $(w_1 : \ldots : w_n : 0)$ are only special due to our choice of an embedding $K^n \hookrightarrow \mathbb{P}^n(K)$. We could choose *any* hyperplane in K^{n+1} not containing 0 as an affine space K^n embedded in $\mathbb{P}^n(K)$. However, for the purposes of this text, it suffices to fix the embedding defined above.

We will mostly deal with affine and projective spaces of dimension $n = 2$. These are called the *affine* or *projective plane* over the respective field K. In particular, we will speak of the *real affine/projective plane* for $K = \mathbb{R}$ and the *complex affine/projective plane* for $K = \mathbb{C}$.

The term “projective” pays tribute to the roots of this kind of geometry. Consider the affine model $\mathbb{R}^3$ of $\mathbb{P}^2(\mathbb{R})$. Imagine the origin to be the eye of an observer and the hyperplane $x_3 = 1$ of finite points to be a transparent canvas. Suppose that rays of light fall from some object beyond the canvas into the eye. Their points of intersection with the canvas yield a perspective drawing, or perspective projection, of the object onto the canvas.

3.1.2 Homogeneous Polynomials

The coordinates of projective points are only defined up to a constant factor λ. This means that we cannot meaningfully evaluate a polynomial $f \in K[x_1,\dots,x_{n+1}]$ at a point $p \in \mathbb{P}^n(K)$. The best we can get is this: Assume that all monomials of f have the same degree. Then we call f *homogeneous,* and it does not depend on the choice of a representative of $p = (p_1 : \dots : p_{n+1})$ whether $f(p_1,\dots,p_{n+1}) = 0$ or not, because $f(\lambda p_1,\dots,\lambda p_{n+1}) = \lambda^{\deg(f)} f(p_1,\dots,p_{n+1})$. This allows us to write $f(p)$ to denote the "value" zero or non-zero of the homogeneous polynomial f at any representative of p.

Given a polynomial $f \in K[x_1,\dots,x_n]$, we can *homogenize* it to

$$F(x_1,\dots,x_{n+1}) = x_{n+1}^{\deg(f)} f(\frac{x_1}{x_{n+1}},\dots,\frac{x_n}{x_{n+1}}) \tag{3.4}$$

which amounts to multiplying every monomial $a_{i_1\dots i_n}x_1^{i_1}\cdots x_n^{i_n}$ of f with the missing power $x_{n+1}^{\deg(f)-i_1-\dots-i_n}$ of x_{n+1}. Then

$$F(p_1 : \dots : p_n : 1) = 0 \iff f(p_1,\dots,p_n) = 0 \tag{3.5}$$

We can also *dehomogenize* a homogeneous polynomial $F \in K[x_1,\dots,x_{n+1}]$ to

$$f(x_1,\dots,x_n) = F(x_1,\dots,x_n,1) \tag{3.6}$$

This reverses the process of homogenization. The converse is not true: Homogenizing the dehomogenization f of F may not yield F again, because whenever x_{n+1} divides F, then $\deg(f)$ is strictly smaller than $\deg(F)$, and (3.4) "forgets" to put back this factor. If x_{n+1} does not divide F, homogenization inverts dehomogenization correctly.

We can write any polynomial $f \in K[x_1,\dots,x_n]$ of degree k uniquely as a sum of its *homogeneous parts*

$$f = f_0 + f_1 + \dots + f_k \quad \text{where } f_d \text{ is homogeneous of degree } d \tag{3.7}$$

We call f_k the *highest-order terms* (HOT) of f. We call the homogeneous part $f_l \neq 0$ of minimal degree l the *lowest-order terms* (LOT) of f. We call f_d the *constant, linear, quadratic,* etc. *part* of f for $d = 0,1,2,\dots$, respectively.

We now turn to the factorization properties of homogeneous polynomials.

Proposition 3.1.1: *Let K be a field, and let $F,G \in K[x_1,\dots,x_{n+1}]$ be homogeneous polynomials with dehomogenizations f and g, respectively.*

(i) The product FG is homogeneous. If x_{n+1} does not divide FG, then FG is the homogenization of fg.

(ii) Any factor of F is homogeneous.

(iii) *Assume $F \neq cx_{n+1}^r$ for any $c \in K$, $r > 0$. F is irreducible in $K[x_1, \dots, x_{n+1}]$ iff f is irreducible in $K[x_1, \dots, x_n]$ and x_{n+1} does not divide F.*

(iv) *The factorization of F into irreducibles consists of the homogenized irreducible factors of f and a power of x_{n+1} (which may be $x_{n+1}^0 = 1$).*

Proof: (i) is easy. For (ii), assume $F = pq$ and observe that the lowest-order terms of F are the product of the lowest-order terms of p and q, respectively [G98, 3.4].

Let us now turn to (iii). If $F = GH$ with proper factors G and H, then either one of them is a power of x_{n+1} or dehomogenizing both yields proper factors of f. Conversely, if $f = gh$ with proper factors g and h, then their homogenizations are proper factors of F.

To see (iv), assume that the maximal power of x_{n+1} has already been extracted from F, then apply (iii) to the unique factorization of f. □

The following corollary [G98, Lemma 3.14] is an important special case of (iv):

Corollary 3.1.2: *Let K be an algebraically closed field, and let $F \in K[x,y]$ be homogeneous of degree $n > 0$. Then*

$$F = \prod_{i=1}^{n} (\beta_i x - \alpha_i y) \quad \textit{for some } \alpha_i, \beta_i \in K, \tag{3.8}$$

where $(\alpha_1 : \beta_1), \dots, (\alpha_n : \beta_n) \in \mathbb{P}^1(K)$ are determined uniquely up to order.

Proof: Let r be the multiplicity of $y = 0x + 1y$ as a factor of F. Then the dehomogenization $f(x) = F(x,1)$ has degree $\deg(f) = n - r$ and decomposes into $n - r$ linear factors $\beta_i x - \alpha_i$. □

F vanishes at $(\alpha : \beta) \in \mathbb{P}^1(K)$ iff $(\beta x - \alpha y)$ is a factor of F, so we can define the *multiplicity* of a zero as the multiplicity of the corresponding linear factor of F and restate the corollary as: If K is algebraically closed, F has exactly n zeroes in $\mathbb{P}^1(K)$, counted with multiplicities.

3.2 Algebraic Curves

3.2.1 Definition of "Algebraic Curve"

Let K be a field, and let $f \in K[x,y]$ be a polynomial. The *zero set* or *vanishing locus* of f is

$$\mathrm{V}(f) := \mathrm{V}_K(f) := \{(x,y) \in K^2 \mid f(x,y) = 0\}. \tag{3.9}$$

Factorization of polynomials corresponds to decomposition of their zero sets:

$$\begin{aligned} f = gh &\implies \mathrm{V}(f) = \mathrm{V}(g) \cup \mathrm{V}(h) \\ g|f &\implies \mathrm{V}(g) \subseteq \mathrm{V}(f) \end{aligned} \tag{3.10}$$

It follows that replacing f by its square-free part does not change $\mathrm{V}(f)$.

The zero set of a non-constant polynomial $f \in K[x,y]$ is, intuitively, a curve. If two polynomials $f,g \in K[x_1,\dots,x_n]$ are *associate* in the sense $f \sim g \;:\Leftrightarrow\; f = \lambda g$ with some unit $\lambda \in (K[x_1,\dots,x_n])^* = K \setminus \{0\}$, their zero sets and divisibility properties are identical. Motivated thereby, we choose to define an *affine algebraic curve* to be an equivalence class $[f]_\sim$ of non-constant associate polynomials $f \in K[x,y]$.

This definition, taken from Gibson [G98], of something that ought to be a point set and not an algebraic object, may be surprising. A quick argument supporting its use is that the algorithm developed in this text works on polynomials and computes geometric properties of their zero sets; relations in the opposite direction are not an issue here.

However, there are deeper reasons in favour of it. Their rigorous treatment is beyond the scope of this text, but an outline can be given: Let K be algebraically closed. From Bezout's Theorem, presented below as Theorem 3.2.4, it can be deduced that a point set in K^2 which is defined as the solution set of a finite system of independent polynomial equations is neither all of K^2 nor a finite subset iff it can be defined as vanishing locus $\mathrm{V}(f) \subseteq K^2$ of a single non-constant polynomial $f \in K[x,y]$. One can deduce further a specialization of Hilbert's famous *Nullstellensatz* [CL$^+$97, 4.1] to this situation, stating that the correspondence between hypersurfaces $\mathrm{V}(f) \subseteq K^2$ and the square-free non-constant polynomials $[f]_\sim \in K[x,y]/\sim$ (up to constants) is bijective. Furthermore, all hypersurfaces in the plane have dimension 1 (in some suitable sense, see [CL$^+$97, Prop. 9.4.2]). That justifies our definition of "algebraic curve" as an equivalence class of non-constant polynomials.

Even though we will mainly be interested in the case where $K = \mathbb{R}$ is not algebraically closed, which implies that the preceding statements on 1-dimensionality and bijective correspondence are false in situations where curves have "too few" real zeroes, it will turn out to be useful to follow this lead.

An affine algebraic curve $[f]_\sim$ can be "lifted" to a *projective algebraic curve* $[F]_\sim$ by homogenizing $f \in K[x,y]$ to $F \in K[x,y,z]$. The common zero set of all representatives of $[F]_\sim$ is

$$\mathrm{V}(F) := \mathrm{V}_K(F) := \{(x:y:z) \in \mathbb{P}^2(K) \mid F(x:y:z) = 0\}. \tag{3.11}$$

$\mathrm{V}(F)$ consists exactly of the affine zero set $\mathrm{V}(f)$ (regarded as a subset of $\mathbb{P}^2(K)$ via (3.3)) and a finite number of points at infinity, viz. the zeroes $(x:y:0)$ of the homogeneous bivariate polynomial $F(x:y:0)$. In this way, an affine algebraic curve defines a projective algebraic curve and its zero set. Proposition 3.1.1 implies that their factorization properties are essentially the same. The statements (3.10) apply analogously.

Let $f \in K[x,y]$ be a non-constant bivariate polynomial. We have carefully distinguished between the polynomial f, the affine algebraic curve $[f]_\sim$ together with its zero set $\mathrm{V}_K(f) \subseteq K^2$, and the projective algebraic curve $[F]_\sim$ defined by f together with its zero set $\mathrm{V}_K(F) \subseteq \mathbb{P}^2(K)$. Whenever $L \supseteq K$ is a superfield of K, one can also consider the zero sets $\mathrm{V}_L(f)$ and $\mathrm{V}_L(F)$. Now that these different notions have been defined, we will allow ourselves the liberty to call the polynomial f an *algebraic curve* or just *curve* and write f for all of these objects, trusting the context to disambiguate the meaning. In particular, we will be considering curves $f \in \mathbb{Q}[x,y]$ and their zero sets $f \subseteq \mathbb{R}^2$ and $f \subseteq \mathbb{C}^2$. Sometimes, we say that the curve f has the *equation* $f(x,y) = 0$.

Unless stated otherwise, we will henceforth assume that a curve is square free. A curve of degree $1,2,3,4\ldots$ is called a *line*, *conic*, *cubic*, *quartic*, etc.

3.2.2 Changing Coordinates

Let K be a field. An *affine change of coordinates* is a bijective map

$$\begin{aligned} A : K^2 &\to K^2 \\ v &\mapsto Mv + b \end{aligned} \tag{3.12}$$

with an invertible matrix $M \in \mathrm{GL}_2(K)$ and a vector $b \in K^2$. The inverse map of $A(v) = Mv + b$ is $A^{-1}(w) = M^{-1}w - M^{-1}b$ and thus again an affine change of coordinates. If $b = 0$, then A is even a *linear change of coordinates*. If $M = 0$, then A is a *translation*.

Given an algebraic curve $f \in K[x,y]$, we can map its zero set $\mathrm{V}(f)$ with an affine change of coordinates A to another set of points $A(\mathrm{V}(f))$. This set is again an algebraic curve: $A(\mathrm{V}(f)) = \mathrm{V}(f \circ A^{-1})$ (where $\circ$ denotes composition of maps), because $f(v) = 0 \;\Leftrightarrow\; (f \circ A^{-1})(A(v)) = 0$. Observe the *contravariant* way in which the inverse of A is needed to transform the equation of the curve. With the convention from §3.2.1, we can write $A(f) = f \circ A^{-1}$ for short.

Consider a translation $A(v) = v + b$. We can express the coefficients of $f \circ A$ using formal derivatives of f, and obtain the *Taylor expansion* of f at b:

$$f(x + b_1, y + b_2) = \sum_{d=0}^{\deg(f)} \sum_{i=0}^{d} \frac{f_{x^i y^{d-i}}(b)}{i!(d-i)!} x^i y^{d-i} \tag{3.13}$$

Most algebraic, analytic, and geometric notions appearing in this text are compatible with suitable changes of coordinates and transform in the more or less obvious fashion. Examples: The total degrees $\deg(f) = \deg(f \circ A)$ agree. The irreducible factors of $f \circ A$ are the transformations $p_i \circ A$ of the irreducible factors p_i of f. The gradient transforms under $A(x) = Mx + b$ by the chain rule as $\nabla(f \circ A)(x) = M^{\mathsf{T}}(\nabla f(A(x)))$. Translating two polynomials $f,g \in K[x,y]$ by (b_1, b_2) translates their resultant w. r. t. y by b_1. Gibson [G98] proves many such statements.

Analogous to the affine case, we define a *projective change of coordinates* as the map $\mathbb{P}^2(K) \to \mathbb{P}^2(K)$ induced by the application of an invertible matrix $M \in \mathrm{GL}_3(K)$ to representatives $(x,y,z) \in K^3$ of points $(x:y:z) \in \mathbb{P}^2(K)$. The projective changes of coordinates include the affine changes of coordinates of the affine plane K^2 embedded in $\mathbb{P}^2(K)$ because

$$\begin{pmatrix} m_{11} & m_{12} & b_1 \\ m_{21} & m_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\ 1 \end{pmatrix} \tag{3.14}$$

As in the affine case, we can transform homogeneous polynomials defining projective algebraic curves in a contravariant fashion which is compatible with the notions of degree, factorization, etc.

The main source of motivations for changing coordinates in the rest of this text is to avoid certain degenerate situations which are not degenerate due to their geometry, but only because of an awkward choice of coordinates. The most prominent example are intersection points that coincide during a resultant computation: Consider two algebraic curves $f,g \in K[x,y]$ with finitely many points of intersection $f \cap g = \{(x_1,y_1), \dots, (x_k,y_k)\}$. When we try to analyze these points, starting from their x-coordinates which are zeroes of $\mathrm{res}(f,g,y)$, it would be helpful if all these x_i were distinct. This is equivalent to the condition that the y-axis shall not be parallel to any of the $\leq \frac{1}{2}k(k-1)$ lines joining (x_i,y_i) and (x_j,y_j) for $i \neq j$. We can achieve this.

Proposition 3.2.1: *Let L be a finite set of lines in K^2. Then all but finitely many choices of $r \in K$ yield a linear change of coordinates*

$$S_r(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} 1 & r \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + ry \\ y \end{pmatrix} \tag{3.15}$$

such that the y-axis is not parallel to any of the lines $S_r(l)$ for $l \in L$.

We call a linear map S_r of this form a *shear* with *shearing parameter r*. Its inverse is $S_r^{-1} = S_{-r}$.

Proof: Clearly, any shear is invertible. Let $l \in L$ be a line. $S_r(l)$ is parallel to the direction $(0,1)$ of the y-axis iff l is parallel to the direction $(-r,1)$. This implies that all but finitely many values of $r \in K$ are permissible. $\square$

Another important property of an algebraic curve $f \in K[x,y]$ is to be y-regular. We can also phrase this in terms of "forbidden directions" for the y-axis. Let $n = \deg(f)$ be the total degree of f. Let f_n be the highest-order terms of f. By Corollary 3.1.2, all factors of f_n over the algebraic closure $\overline{K}$ are linear, and their zero sets in K are either lines through $(0,0)$ (iff the component can be written with coefficients from K) or just $(0,0)$. We have the following equivalences:

$$\begin{aligned} f \text{ is } y\text{-regular} &\iff f = a_{0n}y^n + \dots \text{ with } a_{0n} \neq 0 \\ &\iff x \text{ does not divide } f_n \\ &\iff \text{the } y\text{-axis } x = 0 \text{ is not a subset of } \mathrm{V}(f_n) \end{aligned}$$

Linear changes of coordinates preserve total degree and homogeneity. Hence the highest-order terms of $f \circ S_r$ are $(f \circ S_r)_n = f_n \circ S_r$, so that the zero set of the highest-order terms is well-defined without reference to a specific coordinate system. This proves the following corollary to Proposition 3.2.1.

Corollary 3.2.2: *Let L be a finite set of lines in K^2, and let $C \subseteq K[x,y]$ be a finite set of algebraic curves. Then all but finitely many choices of a shearing parameter $r \in K$ yield a shear S_r such that $f \circ S_{-r}$ is y-regular for all $f \in C$ and that the y-axis is not parallel to any of the lines $S_r(l)$ for all $l \in L$.*

There is a more intuitive way to understand the significance of f_n. The polynomial x divides f_n iff the homogenization F of f vanishes at $(0:1:0)$. Then there are tangents to F at $(0:1:0)$ (as we will see later on). If such a tangent has a defining polynomial in $K[x,y]$ (and not just over the algebraic closure $\overline{K}$), then this tangent is visible in the affine plane as a vertical asymptote of f. We will introduce vertical asymptotes formally in §3.2.5.

3.2.3 Components, Multiplicities, and Intersections

Let $f \in K[x,y]$ be an algebraic curve (not necessarily square-free), and let $\overline{K}$ be the algebraic closure of K. The polynomial $f \in \overline{K}[x,y]$ factors essentially uniquely into coprime irreducible factors p_i with multiplicities $e_i > 0$ as in (2.3). By (3.10), this corresponds to a decomposition of the zero set f into subsets p_i which cannot be decomposed any further in this fashion. We call each p_i an *(irreducible) component* of f with *multiplicity* e_i.

Note that this definition of irreducible component may take us outside the field K of coefficients. As an example, consider $f = y^2 - 2x^2 = (y + \sqrt{2}x)(y - \sqrt{2}x)$ over the field $K = \mathbb{Q}$. A finer decomposition into irreducible factors than the one obtained over the algebraic closure $\overline{K}$ is not possible, because the coefficients of the factors are solutions to algebraic equations over K and hence contained in $\overline{K}$.

Let $v \in \overline{K}^2$ be a point. The smallest $m \geq 0$ such that there is an m-th partial derivative of f not vanishing at v is called the *multiplicity* $\mathrm{mult}(v; f) := m$ of v on f. In particular $\operatorname{mult}(v;f) > 0 \Leftrightarrow v \in f$. By (3.13), this is equivalent to defining the multiplicity as the degree of the lowest-order terms of $f(x+v_1, y+v_2)$. From this characterization, it is immediate that

$$\operatorname{mult}(v;gh) = \operatorname{mult}(v;g) + \operatorname{mult}(v;h). \tag{3.16}$$

Multiplicity of a point is invariant under coordinate changes.

We want to investigate geometric properties of the curve f using multiplicities of points. Multiplicities of components are irrelevant for the zero set, hence we simplify matters and demand f to be square-free from now on. Then almost all points on f have multiplicity 1. (We will prove this later as Corollary 3.2.9.) We call them *regular*. The remaining points of multiplicity ≥ 2 are called *singular*.

Now we turn to the intersection of two algebraic curves. Again, a notion of multiplicity will be important. Imagine intersecting a parabola $y = x^2 + c$ with the x-axis $y = 0$ in $\mathbb{C}^2$. There are two distinct points of intersection $(\pm\sqrt{c}, 0)$ for $c \neq 0$, but there is only one for $c = 0$. Looking at this situation in $\mathbb{R}^2$ indicates that the case $c = 0$ is special because then the x-axis touches the parabola instead of cutting through it, which forms a limiting situation between two distinct real and two complex-conjugate intersections. We will make these ideas precise in the sequel, and we start with the following definition.

Let $f, g \in K[x,y]$ be two coprime algebraic curves. By Proposition 2.3.4, they have only a finite number of points in common. According to Corollary 3.2.2, we can choose coordinates such that f, g are y-regular and that no two intersection points of f and g have the same x-coordinate. Then there is a bijective correspondence between the zeroes of $r := \operatorname{res}(f,g,y)$ and the points of $f \cap g$ by Proposition 2.3.3. We define the *intersection multiplicity* $\operatorname{mult}(v;f,g)$ of f and g at an intersection point $v \in f \cap g$ as the multiplicity of its x-coordinate v_1 as a root of r, and as 0 for $v \notin f \cap g$. This definition is independent of the choice of coordinates, but that is not obvious, see [CL$^+$97, 8.7] [BK86, 6.1]. This definition extends to projective algebraic curves by dehomogenization (if necessary after a projective change of coordinates to make sure that no intersections are lost at infinity).

The intersection multiplicity of f with a line g has a straightforward interpretation:

Proposition 3.2.3: *Let $g \in K[x,y]$ be a line, and let $f \in K[x,y]$ be an algebraic curve not containing g. Let $t \mapsto (at+v_1, bt+v_2)$ be a parametrization of g. Then $\operatorname{mult}(v;f,g)$ is the multiplicity of $t = 0$ as a zero of $f(at+v_1, bt+v_2) \in K[t]$.*

For a proof, see [G98, Lemma 14.6].

The most fundamental result on the intersection of plane algebraic curves is this:

Theorem 3.2.4: *(Bezout's Theorem)*
Let K be an algebraically closed field, and let $F, G \in K[x,y,z]$ be two coprime projective algebraic curves. Then they have exactly $\deg(F)\deg(G)$ intersection points in $\mathbb{P}^2(K)$, counted with multiplicities.

Actually, our way of defining the intersection multiplicity makes it easy to prove the theorem, as its statement is equivalent to $\deg(f)\deg(g) = \deg(\mathrm{res}(f,g,y))$ for dehomogenizations in a suitable coordinate system, see [G98, 14.4] [CL$^+$97, 8.7] [BK86, 6.1]. An axiomatic approach to intersection multiplicities is followed in Fulton [F69], together with an advanced formulation based on local rings instead of resultants.

From now on, we will focus mainly on the case $K = \mathbb{R}$. The immediate consequence of Bezout's Theorem for the affine real plane is given by the following corollary.

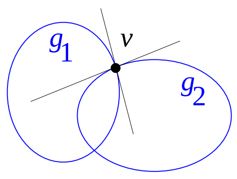
Corollary 3.2.5: *Let K be a field, and let $f,g \in K[x,y]$ be two coprime affine algebraic curves. Then they have at most* $\deg(\mathrm{res}(f,g,y))$ *intersection points in K^2, counted with multiplicities, and* $\deg(\mathrm{res}(f,g,y)) \leq \deg(f)\deg(g)$.

3.2.4 Tangents and Singularities

Let v be a point on the algebraic curve $f \in K[x,y]$. Translate the coordinate system such that $v = 0$. Then the lowest-order terms f_m of f form a homogeneous polynomial of degree $m = \mathrm{mult}(v; f) \geq 1$ and factorize over $\overline{K}$ into exactly m linear factors $l_1, \ldots, l_m$ (counted with multiplicities), called the *tangents* of f at v. If one of the tangents is the vertical line x, we say the point v is *vertical* (is a *vertical point*, a *vertical singularity*, etc. of f.)

For $K = \mathbb{R}$ and $m = 1$, it is easy to see that $f_1 = l_1$ is the tangent of f at v in the sense of differential geometry: Change coordinates such that $v = 0$. Then $f = ax + by + \ldots$, and $\nabla f(v) = (f_x(v), f_y(v)) = (a,b)$ is the *normal vector* of f at $v = 0$, yielding the tangent $ax + by = 0$. The analogous statement holds for $K = \mathbb{C}$.

It is also easy to extend this further to singularities arising from the intersection of components: Let $g_1, \ldots, g_m$ be curves, and let v be a regular point on all of them. Furthermore, let h be a curve not containing v. Set $f = g_1 \cdots g_m h$. Then v is a point of multiplicity m on f, and the m tangents of f at v are precisely the tangents of $g_1, \ldots, g_m$ at v, because for $v = 0$ the lowest-order terms of f are the product of the constant LOT of h and the linear LOT of the g_i. It is obvious how the curve f locally consists of exactly m "branches", viz. the curves g_i, that meet in v.



It is much harder, though, to understand singularities that already exist in an irreducible curve, and how they also permit a sensible notion of branches meeting in the singularity. We can imagine the situation for $K = \mathbb{C}$ to be as follows: The lowest powers $x^i y^{m-i}$ in f dominate f close to $v = 0$, so that the zero set of f can be "approximated" around v by the zero set of f_m, viz. the union of the tangents $l_1, \ldots, l_m$. In particular, we expect m "branches" of f to meet at v, each one having one of the tangents of f at v as its tangent. When we try this with a real algebraic curve $f \in \mathbb{R}[x,y]$ we face additional sources of confusion: Some of the m complex tangents may not have defining polynomials with real coefficients, but even if some tangent has real coefficients, it may occur twofold, and the two branches it belongs to may be complex conjugates (at least on one side of the singularity). We will meet an example (a "cusp") in §3.3.4.

The reader is referred to the book of Brieskorn and Knörrer [BK86] for a proper yet heavyweight treatment of singularities. For the limited class of curves relevant to this text, a phenomenology of singularities will be given in §3.3.4.

We conclude with a proposition on the relevance of multiplicities and tangents in intersections [G98, 14.5].

Proposition 3.2.6: *(Multiplicity Inequality)*
Let K be a field, and let $v \in K^2$ be a point. Let $f, g \in K[x,y]$ be two coprime algebraic curves. Then

$$\operatorname{mult}(v; f, g) \geq \operatorname{mult}(v; f) \operatorname{mult}(v; g), \tag{3.17}$$

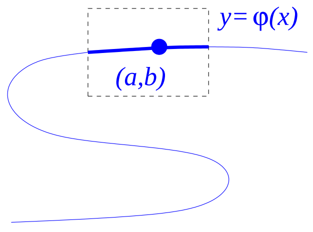
and equality holds iff f and g have no common tangent at v.

The effects of a common tangent in an intersection of regular points will be quantified further by Proposition 3.2.11.

The intersection of a line g with a regular point v of f has multiplicity 1, except if g is the tangent of f at v. If g indeed is the tangent of f at v, then the intersection multiplicity is at least 2. If it is larger than 2, v is called a *flex* of f. A curve can only have a finite number of flexes (because they can be characterized as intersections with a coprime curve, called the *Hessian*), see [G98, 13.1].

3.2.5 Implicit Functions and Arcs

In §2.2.1, we distinguished flat and hierarchical views on polynomials. So far we have looked at algebraic curves from a flat point of view, regarding points as atomic entities. We will now turn to a hierarchical point of view, considering x-coordinates as free and y-coordinates as dependent quantities. The class of polynomial functions is too restrictive to support this view, so we consider analytic functions instead. Recall that a function $f : D \to K$, where $K = \mathbb{R}$ or $K = \mathbb{C}$, is called *analytic* if it can be expressed as a convergent power series in a neighbourhood of every point of its domain $D \subseteq K^n$. This trivially includes polynomial functions on $D = K^n$.



Theorem 3.2.7: *(Implicit Function Theorem for Analytic Functions)*
Let K be $\mathbb{R}$ or $\mathbb{C}$. Let $f : D \to K$ be an analytic function of some open set $D \subseteq K^2$, and let $(a,b) \in D$ be such that $f(a,b) = 0$ and $f_y(a,b) \neq 0$. Then there are neighbourhoods U_a of a and U_b of b as well as a uniquely determined analytic function $\varphi : U_a \to U_b$ such that

$$f(x,y) = 0 \iff y = \varphi(x) \quad \textit{for all } (x,y) \in U_a \times U_b. \tag{3.18}$$

Proof: A proof for $K = \mathbb{C}$ can be found in [BK86, p. 348].

For $K = \mathbb{R}$, apply the implicit function theorem for real continuously differentiable functions [L97, XVIII.4] [K97, 3.4]. Its statement is completely analogous to the theorem above, except that it deals with real C^1 functions instead of analytic functions, yielding an implicit function φ. It remains to conclude that φ is not just C^1 but analytic under the premise that f is analytic. To see this, observe that the analytic function f on $D \subseteq \mathbb{R}^2$ has an analytic continuation $\tilde{f}$ on an open set $\tilde{D} \subseteq \mathbb{C}^2$ containing $D \subseteq \tilde{D}$ to which we can apply the above theorem for $K = \mathbb{C}$. The resulting implicit function $\tilde{\varphi} : U_a \to U_b$ agrees with φ on all points of $U_a \cap \mathbb{R}$, so that $\varphi : U_a \cap \mathbb{R} \to \mathbb{R}$ is analytic. □

Let $f \in \mathbb{R}[x,y]$ be a y-regular square-free algebraic curve. Then f_y is also y-regular. We will now analyze the geometry of f, using f_y and the Implicit Function Theorem. The Implicit Function Theorem is applicable at almost every point of f:

Proposition 3.2.8: *Let $f \in K[x,y]$ be a y-regular square-free algebraic curve. Then f has only finitely many points in common with f_y.*

Proof: By Corollary 3.2.5, it suffices to show that f and f_y are coprime. Assume to the contrary that there is a common factor d of f and f_y with $\deg(d) > 0$. This factor d is y-regular by Proposition 2.2.1. Proposition 2.2.10 implies that d^2 divides f in $Q(\mathbb{R}[x])[y]$. Being y-regular, d^2 is primitive in $\mathbb{Q}[x][y]$. Thus d^2 divides f already in $\mathbb{R}[x,y]$ by Corollary 2.2.9, contradicting the squarefreeness of f. □

Corollary 3.2.9: *Let $f \in K[x,y]$ be a square-free algebraic curve. Then f has only a finite number of singular points.*

Proof: Choose coordinates such that f is y-regular. Every singular point is an intersection of f and f_y. □

Now we turn to basic topological properties of the point set $f \subseteq \mathbb{R}^2$. It is the preimage of $\{0\}$ under the continuous map f and hence (topologically) closed. The subset $f \setminus f_y$ is open in f. By pasting together the local parametrizations from the Implicit Function Theorem, we see that the connected components of $f \setminus f_y$ are parametrized curves

$$\begin{array}{rcl} \gamma_i :]l_i, r_i[& \to & \mathbb{R}^2 \\ x & \mapsto & (x, \varphi_i(x)) \end{array} \tag{3.19}$$

with some analytic functions φ_i and interval boundaries l_i, r_i from the *two-point compactification* $\mathbb{R} \cup \{\pm\infty\}$ of $\mathbb{R}$. In particular, every connected component of $f \setminus f_y$ is a C^∞-manifold of dimension 1. We call the (topological) closure $A_i := \mathrm{cl}(\gamma_i(]l_i, r_i[))$ of each such component an *arc* of f.

Since $\gamma_i(x) = (x, \varphi_i(x))$, a boundary point of A_i must have an x-coordinate which is a boundary point of $]l_i, r_i[$. We can thus speak of a *left endpoint* at $x = l_i$ and a *right endpoint* at $x = r_i$, if they exist within $\mathbb{R}^2$. In the sequel, we are sloppy and also talk of left and right endpoints *at infinity*, denoting the absence of an endpoint in $\mathbb{R}^2$. (This terminology will be justified during the case distinction below.)

The topology of f is determined by the way in which the arcs are connected at their endpoints. Four things can happen at an endpoint of an arc $A = \mathrm{cl}(\gamma(]l, r[))$. We consider the right endpoint; the left endpoint behaves analogously.

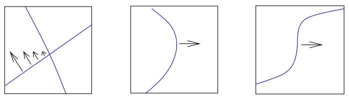
1. **Infinity:** If $r = +\infty$, the right endpoint does not exist in $\mathbb{R}^2$. Instead, it is an endpoint at infinity, and it does not link A to other arcs. We say A *extends to infinity* (at its right end).

This is the only way an endpoint at infinity can arise in the y-regular curve f. If $r \in \mathbb{R}$ is finite, then the right endpoint is not at infinity. To prove this by way of contradiction, assume otherwise. We must have $s := \lim_{x\to r} \varphi(x) = \pm\infty$, or else $(r, s) \in \mathbb{R}^2$ would be a boundary point of A. This means that

$$(x : \varphi(x) : 1) = \left(\frac{x}{\varphi(x)} : 1 : \frac{1}{\varphi(x)} \right) \to (0 : 1 : 0) \quad \text{for } x \to r,$$

or in other words: There is a *vertical asymptote* at $x = r$. By continuity, this implies $F(0 : 1 : 0) = 0$ for the homogenization F of f, so that x divides the highest-order terms of f, contradicting y-regularity (cf. the remark following Corollary 3.2.2) and proving the claim.

This settles the case of endpoints at infinity. In the remaining cases, there is a right endpoint $(r, s) \in f \cap f_y \subseteq \mathbb{R}^2$ with $s = \lim_{x\to r} \varphi(x)$, and we classify such a finite endpoint as singularity, x-extremality, or vertical inflexion (see figure).



2. **Singularity:** If $f_x(r, s) = f_y(r, s) = 0$, then (r, s) is a singular point, and an arbitrary finite number of arcs (maybe zero) meet there; see §3.2.4.

The remaining cases are those with $f_x(r, s) \neq 0$, so that (r, s) is a regular point of f whose unique tangent l is the vertical line $l = x - r$. We obtain $\mathrm{mult}((r, s); f, l)$ using Proposition 3.2.3 by substituting the parametrization $t \mapsto (r, s + t)$ of l into f. With (3.13), the resulting polynomial can be written as

$$f(r + 0, s + t) = f(r, s) + f_y(r, s)t + \frac{1}{2} f_{yy}(r, s)t^2 + \ldots \tag{3.20}$$

and the multiplicity m of its zero $t = 0$ is exactly $m = \mathrm{mult}((r,s); f, l)$. Recall that $f(r,s) = f_y(r,s) = 0$, hence $m \geq 2$.

Since $\nabla f(r,s) = (f_x(r,s), f_y(r,s))^{\mathsf{T}} \neq 0$, shearing the coordinate system with any non-zero shearing parameter (or just swapping x and y) will make f_y non-zero at (r,s). By applying the Implicit Function Theorem in this changed coordinate system, we see that f is a 1-dimensional C^∞-manifold around (r,s), just as around any other regular point. Furthermore, Corollary 3.2.12 (to be proved below) implies that the curve f changes sides with l at (r,s) iff m is odd. (One could also prove this special case directly.)

After this prelude, we can distinguish two more cases:

3. **x-Extremality:** Let $f_y(r,s) = 0$, $f_x(r,s) \neq 0$, and $f_{yy}(r,s) \neq 0$. Then (r,s) is a regular point with a vertical tangent l, and (3.20) implies $\mathrm{mult}((r,s); f, l) = 2$, so that f lies on just one side of l near (r,s). We call such a point (r,s) an *x-extreme point* of f. An x-extreme point joins two arcs smoothly.

   Since we are investigating a right endpoint, f lies to the left of l, and we can call (r,s) *right x-extreme* or *x-maximal*. For a left endpoint, the corresponding terms are *left x-extreme* or *x-minimal*.

   The condition $f_{yy}(r,s) \neq 0$ guarantees that f_y is regular at (r,s) and does not share the vertical tangent l with f. Hence $\mathrm{mult}((r,s); f, f_y) = 1$ by Proposition 3.2.6.

4. **Vertical Inflexion:** Let $f_y(r,s) = 0$, $f_x(r,s) \neq 0$, but $f_{yy}(r,s) = 0$. Then (r,s) is a regular point with a vertical tangent l and $m = \mathrm{mult}((r,s); f, l) \geq 3$, so that (r,s) is a vertical flex. A vertical flex joins two arcs smoothly.

   Bezout's theorem implies $m \leq \deg(f)$, so that a vertical flex cannot occur for lines and conics, and must have $m = 3$ for cubic curves.[1]

   By inverting the corresponding argument made above for x-extreme points, we obtain $\mathrm{mult}((r,s); f, f_y) \geq 2$ (see also Proposition 3.3.3).

This does not only describe the situations at endpoints of arcs; the cases 2–4 also classify all elements of $v \in f \cap f_y$ both in $\mathbb{R}^2$ and in $\mathbb{C}^2$, since they form a complete case distinction on the values of f_x and f_{yy} at any v. By Proposition 3.2.8, $f \cap f_y$ is finite, and only a finite number of arcs meet at each $v \in f \cap f_y$, so that the number of arcs is also finite.

Finally, let us observe that the equality (3.20) holds for arbitrary points (r,s), thereby proving the following useful characterization of $f \cap f_y$ (which could also be obtained along the lines of §2.2.3):

[1]This means that there is no need to decide whether vertical flexes with even m ought to be called x-extreme or not if only curves up to degree 3 are under discussion.

Corollary 3.2.10: *Let $f \in K[x,y]$ be a y-regular algebraic curve, and let $r,s \in K$. Then the following two statements are equivalent:*

(i) The point (r,s) is an intersection of f and f_y.

(ii) The polynomial $f(r,y) \in K[y]$ has a multiple root at $y = s$.

3.2.6 Intersections within Arcs

Let us turn to intersections of arcs of two distinct curves. The intersection multiplicity defined in §3.2.3 blends well with implicit functions for intersections in the interior of arcs.[2]

Proposition 3.2.11: *Let K be $\mathbb{R}$ or $\mathbb{C}$. Let $f,g \in K[x,y]$ be two coprime y-regular algebraic curves. Let $v \in K^2$ be a point such that $f(v) = 0$ and $g(v) = 0$ but $f_y(v) \neq 0$ and $g_y(v) \neq 0$. Then* $\operatorname{mult}(v; f,g)$ *is the smallest exponent d for which the coefficients of x^d in the implicit power series of f and g around v disagree.*

Proof: We can restrict attention to the general case $K = \mathbb{C}$. We choose coordinates such that $v = (0,0)$, that no two points in $f \cap g$ have the same x-coordinate, and that no point in $(f \cap f_y) \cup (g \cap g_y)$ has an x-coordinate equal to $v_1 = 0$. Assume further that f and g are monic.

The polynomial $f(0,y)$ has precisely $n := \deg(f)$ zeroes, and by applying the implicit function theorem to each of these, we obtain n convergent power series $\varphi_1(x),\ldots,\varphi_n(x)$ such that the identity $f(x,y) = \prod_{i=1}^{n}(y - \varphi_i(x))$ holds in $\mathbb{C}[y]$ for all $x \in U$ in some neighbourhood U of 0. This implies that the identity also holds as an identity of formal power series in the indeterminates x and y.

Doing the same for g, which has degree $m := \deg(g)$, we obtain the following two equalities of power series:

$$f(x,y) = \prod_{i=1}^{n}(y - \varphi_i(x)) \qquad\qquad g(x,y) = \prod_{j=1}^{m}(y - \chi_j(x))$$

We choose indices such that $i = 1$ and $j = 1$ correspond to the arcs intersecting at v, that is $\varphi_1(0) = \chi_1(0) = 0$.

By Proposition 2.3.2, the resultant used in the definition of intersection multiplicity equals

$$r := \operatorname{res}(f,g,y) = \prod_{i=1}^{m}\prod_{j=1}^{n}(\varphi_i - \chi_j)$$

The multiplicity of 0 as zero of a power series is defined as the lowest exponent d of a power x^d with non-zero coefficient. This extends the corresponding notion

[2]The proof is based on an idea that the author learned from Nicola Wolpert.

of multiplicity for polynomials. In a product of power series, the multiplicities of zeroes add up. By our choice of coordinates, the only pair (i, j) for which $\varphi_i - \chi_j$ vanishes at 0 is $(1,1)$. Hence the multiplicity of $x = 0$ as a zero of $\operatorname{res}(f,g,y)$ is d. This proves the claim in the chosen coordinate system.

To transport this back into the original coordinate system, we need invariance of the number of agreeing coefficients under changes of coordinates that do not make f_y vanish at v. We do not prove this here, and the algorithm will only depend on the statement of the proposition in cases where the conditions on the coordinate system other than $v = (0,0)$ are already fulfilled. □

Corollary 3.2.12: *In the situation of Proposition 3.2.11 for $K = \mathbb{R}$, the two arcs of f and g intersecting at a point v change sides iff* $\operatorname{mult}(v; f,g)$ *is odd.*

Proof: Let $m := \operatorname{mult}(v; f,g)$. With notation analogous to the proof of the proposition, consider the analytic function $\delta : U \to \mathbb{R}$, $x \mapsto \varphi_1(x) - \chi_1(x)$. For $x \to 0$, it is dominated by its lowest-degree term cx^m which changes sign at 0 iff m is odd. □

3.3 Cubic Curves

Let us now come to the specific class of curves to be handled by the algorithm developed in this text: algebraic curves $f \in \mathbb{Q}[x,y]$ with $\deg(f) \leq 3$, that is cubics, conics, and lines with rational coefficients; and the geometry of their zero sets $V_{\mathbb{R}}(f)$ in the real affine plane.

3.3.1 Algebraic degrees of special points

Points of interest on an individual curve $f \in \mathbb{Q}[x,y]$ (cf. §3.2.5) and intersections of two such curves (cf. §2.3.2) in $\mathbb{C}^2$ have coordinates that are algebraic numbers. The same holds for the coefficients of the components of f. For algorithmic purposes, it is useful to understand the degrees of these algebraic numbers.

Let us look at the points of $f \cap f_y$. We start with a result that allows us to restrict attention to x-coordinates.

Proposition 3.3.1: *Let $f \in \mathbb{Q}[x,y]$ be a y-regular cubic curve of degree* $\deg(f) \leq 3$. *Let $(\vartheta,\eta), (\vartheta,\eta') \in V_{\mathbb{C}}(f) \cap V_{\mathbb{C}}(f_y)$ be two points at the same x-coordinate. Then $\eta = \eta'$ and $\eta \in \mathbb{Q}(\vartheta)$.*

Proof: Consider $\varphi(y) := f(\vartheta,y) \in \mathbb{Q}(\vartheta)[y]$. By Corollary 3.2.10, η and η' are multiple roots of φ. Since $\deg(\varphi) = \deg(f) \leq 3$, factoring φ by multiplicities as in (2.6) yields exactly one factor χ of a multiplicity larger than 1, and χ must be linear. Hence $\chi(\eta) = \chi(\eta') = 0$ implies $\chi = y - \eta = y - \eta' \in \mathbb{Q}(\vartheta)[y]$, proving the claim. □

This means that an x-coordinate ϑ determines the corresponding y-coordinate η uniquely and bounds its algebraic degree. With Proposition 2.3.3, it follows that the points (ϑ,η) of $f \cap f_y$ are in bijective correspondence to the zeroes ϑ of $r := \mathrm{res}(f, f_y, y)$. By Bezout's Theorem, $\deg(r) \leq 6$, so 6 is a trivial upper bound for the algebraic degrees of coordinates of all points in $f \cap f_y$. But we can do better.

Proposition 3.3.2:
Let $f \in \mathbb{Q}[x,y]$ be a y-regular algebraic curve of degree $\deg(f) \leq 3$.

The number of x-extreme points of f in $\mathbb{C}^2$ is an upper bound on the algebraic degrees of their coordinates. It is at most 6.

The number of singular points and vertical flexes of f in $\mathbb{C}^2$ is an upper bound on the algebraic degrees of their coordinates. It is at most 3.

Proof: Factor $r := \mathrm{res}(f, f_y, y) = r_1 r_2^2 r_3^3 \cdots$ by multiplicities to obtain a square-free polynomial $r_1 \in \mathbb{Q}[x]$ containing all simple zeroes and a square-free polynomial $s = r_2 r_3 \cdots \in \mathbb{Q}[x]$ containing all multiple zeroes of r. According to §3.2.5, the zeroes of r_1 correspond to the x-extreme points of f whereas the zeroes of s correspond to singularities and vertical flexes. Observe $\deg(s) \leq \frac{1}{2}\deg(r) = 3$. The existence of these rational polynomials vanishing at the respective x-coordinates proves the claim. □

This statement is not surprising in the light of the more general Proposition 2.4.3, see Section A.2. We prefer the less sophisticated proof above because our algorithm will follow its explicit construction of the defining polynomials.

The mixture of vertical flexes and singularities is a nuisance. A flex can never be an algebraic conjugate of a singularity: Using the method of §2.4.2 we may replace the polynomial s from the proof by some rational factor of s, retaining precisely those zeroes ϑ of s for which $f_x(\vartheta,\eta) = 0$, proving that the number of singularities of f in $\mathbb{C}^2$ bounds their algebraic degrees.

In our algorithm, we will use another method to exclude vertical flexes. It will be helpful to know which square-free factor of $\mathrm{res}(f, f_y, y)$ contains them. (In §3.2.5 we could only state a lower bound on the degree.)

Proposition 3.3.3: *Let $f \in \mathbb{C}[x,y]$ be a y-regular cubic curve with a vertical flex $v = (\vartheta,\eta) \in \mathbb{C}^2$. Then the multiplicity of ϑ as a zero of* $\mathrm{res}(f, f_y, y)$ *is exactly 2.*

Proof: After a suitable translation, we may assume $v = 0$. We may also assume $\ell(f) = 1$. By §3.2.5 we have $f(v) = f_y(v) = f_{yy}(v) = 0$ and $f_x(v) \neq 0$, so that (3.13) yields

$$\begin{aligned} f(x,y) &= y^3 + a_1 xy^2 + b_2 x^2 y + c_3 x^3 + b_1 xy + c_2 x^2 + c_1 x \\ f_y(x,y) &= 3y^2 + 2a_1 xy + b_2 x^2 + b_1 x \end{aligned}$$

with $c_1 \neq 0$, which implies

$$\operatorname{res}(f, f_y, y) = \begin{vmatrix} 1 & a_1x & b_2x^2+b_1x & c_3x^3+c_2x^2+c_1x & \\ & 1 & a_1x & b_2x^2+b_1x & c_3x^3+c_2x^2+c_1x \\ 3 & 2a_1x & b_2x^2+b_1x & & \\ & 3 & 2a_1x & b_2x^2+b_1x & \\ & & 3 & 2a_1x & b_2x^2+b_1x \end{vmatrix}$$

where we can extract a factor of x in the last two columns, leading to

$$\ldots = \begin{vmatrix} 1 & a_1x & b_2x^2+b_1x & c_3x^2+c_2x+c_1 & \\ & 1 & a_1x & b_2x+b_1 & c_3x^2+c_2x+c_1 \\ 3 & 2a_1x & b_2x^2+b_1x & & \\ & 3 & 2a_1x & b_2x+b_1 & \\ & & 3 & 2a_1 & b_2x+b_1 \end{vmatrix} \cdot x^2$$

Hence the multiplicity of 0 as a zero of $\operatorname{res}(f, f_y, y)$ is at least 2, but it is no larger, since substituting $x = 0$ into the determinant above yields

$$\begin{vmatrix} 1 & & & c_1 & \\ & 1 & & b_1 & c_1 \\ 3 & & & & \\ & 3 & & b_1 & \\ & & 3 & 2a_1 & b_1 \end{vmatrix} = 3 \cdot c_1 \cdot 3 \cdot c_1 \cdot 3 = 27c_1^2 \neq 0.$$

To see the equality, expand this determinant successively by the third, fifth, second and fourth column, in this order. □

This ends our theoretical treatment of vertical flexes. Before we consider the algebraic degrees of components, we must first understand how a cubic can factor and how this gives rise to singularities.

3.3.2 Complex Projective Classification of Cubics

In order to classify lines, conics, and cubics with respect to number and kind of their singularities, using the textbook classification of conics and cubics up to complex projective equivalence [G98] turns out to be a useful basis. We reproduce this classification without proofs below. Although we are interested in cubics in the affine real plane $\mathbb{R}^2$, we are happy with this coarse classification, and especially its brevity: For real affine cubics, Isaac Newton found 72 different "species" and overlooked six others [BK86, p. 92] [B98, p. 128].

One says that two curves f and g are *projectively equivalent* if there is a projective change M of coordinates such that their homogenizations satisfy $F = G \circ M$. Note that this definition depends on the field K, because the larger K is, the more transformations there are. We will henceforth use the algebraically closed field $K = \mathbb{C}$. Projective equivalence preserves the number of singularities as well as the multiplicity and (in)equality of tangents at a singularity.

Recall that we are only interested in square-free curves.

Lines and Conics

Let $f \in \mathbb{C}[x,y]$ be a line, i. e. $\deg(f) = 1$. It is irreducible, and it possesses no singularities. All lines are projectively (even affinely) equivalent.

Let $f \in \mathbb{C}[x,y]$ be a conic. It is either irreducible or a pair of two distinct lines. An irreducible conic possesses no singularities. All irreducible conics are projectively equivalent.

By (3.16) in conjunction with Bezout's Theorem, a line pair $f = g_1 g_2$ has a unique singularity in the projective plane, viz. the unique intersection point v of its two components g_1 and g_2, which may or may not lie at infinity. The singularity v is a *node*, meaning that it has multiplicity 2 and that its two tangents (in this case, the two lines themselves) are distinct. All line pairs are projectively equivalent.

Irreducible Cubics

Let $f \in \mathbb{C}[x,y]$ be an irreducible cubic. It has at most one singular point in the projective plane, which may or may not lie at infinity. If f has a singular point (maybe at infinity), it is called *singular*, otherwise *non-singular*.

The unique singularity v of a singular irreducible cubic f has multiplicity 2, so that there are exactly two tangents at v. They may either be distinct, in which case v is a node, or equal, in which case v is called a *cusp*. All irreducible cubics with a node are projectively equivalent, and all irreducible cubics with a cusp are projectively equivalent.

The non-singular cubics are not pairwise affinely equivalent; instead, there is an infinite family of normal forms representing all equivalence classes.

Cubics with several components

Let $f \in \mathbb{C}[x,y]$ be a cubic that is not irreducible. Then it has two components (a line and an irreducible conic) or three components (three lines), corresponding to the two possible decompositions of $\deg(f) = 3$ as $3 = 2 + 1$ or $3 = 1 + 1 + 1$.

Since lines and irreducible conics have no singularities, by (3.16) all singularities of f are intersections of its components. Some of these singularities may lie at infinity.

Assume $f = gh$ consists of a line g and an irreducible conic h. By Bezout's Theorem, g and h intersect in exactly two points without common tangent or in exactly one point with a common tangent. In the former case, g is a *chord* to h, and the two intersections are nodes. In the latter case, g is a tangent to h at the unique intersection point, and the resulting intersection is a cusp.

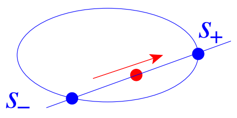
Now assume $f = g_1 g_2 g_3$ consists of three distinct lines. If these three lines have a common point $v \in g_1 \cap g_2 \cap g_3$ (maybe at infinity), then v is the unique singularity of f with $\operatorname{mult}(v; f) = 3$ (a *triple point*) with the three distinct tangents g_1, g_2, and g_3, and we call f a *star*. Otherwise, the three lines form a *triangle*, i. e. for all choices $\{i, j, k\} = \{1, 2, 3\}$ of three distinct indices there is exactly one intersection point $s_k \in g_i \cap g_j$, $s_k \notin g_k$; and s_k (which may lie at infinity) is a node of f with the two distinct tangents g_i and g_j. We call s_1, s_2, s_3 the *vertices* of the triangle f.

3.3.3 Components of rational conics and cubics

Let us draw conclusions from §3.3.2 for curves $f \in \mathbb{Q}[x, y]$ with rational coefficients and more than one component.

First, we consider the case of a cubic $f \in \mathbb{Q}[x, y]$ consisting of a line g and an irreducible conic h with equations $g, h \in \mathbb{C}[x, y]$. As before, distinguish the cases of a chord g and a tangent g. (Note that the term chord, defined with reference to the complex plane, includes lines g that have no real intersections with h.) In both cases, we will prove that $g, h \in \mathbb{Q}[x, y]$. We assume that all intersections of g and h lie in the affine plane. In the exceptional case that this does not hold, a projective change of coordinates with a rational matrix gets us into this situation without changing rationality. Similarly, we may assume that f and thus also g and h are y-regular.

So let g be a chord. The two intersections of g and h are the two singularities of f. If they are rational, then g is a line through two rational points and hence rational. Otherwise, they are algebraic of degree 2 by Proposition 3.3.2 and conjugate by Proposition 2.4.3. Any quadratic polynomial can be solved with one square root, and by Proposition 3.3.1 the same square root suffices for both coordinates, so that the two singularities are $s_+ = (a + b\sqrt{D}, c + d\sqrt{D})$ and $s_- = (a - b\sqrt{D}, c - d\sqrt{D})$ for some $a, b, c, d, D \in \mathbb{Q}$. Then $\frac{1}{2}(s_+ + s_-) = (a, c)$ is a point on g and $\frac{1}{2i}(s_+ - s_-) = (b, d)$ is the direction of g, hence $g \in \mathbb{Q}[x, y]$ and also $h = f/g \in \mathbb{Q}[x, y]$.



Now let g be a tangent. Then the unique singularity v of f must be rational by Proposition 3.3.2. According to §3.3.2, v is a cusp so that g is obtained in $\mathbb{Q}[x, y]$ as the square-free factor g of multiplicity 2 of the quadratic part of the translated curve $f(x + v_1, y + v_2) \in \mathbb{Q}[x, y]$. Hence again $g, h \in \mathbb{Q}[x, y]$.

These results are useful in the generation of test data for our algorithm: It incurs no limitation if test cases for cubics consisting of line and conic are generated as products of a rational line and a rational conic. To enforce irrational intersections, one can use a conic without rational points like those constructed in Section A.3.

Let us now consider the case of a curve $f \in \mathbb{Q}[x,y]$ that is a conic $f = g_1g_2$ or a cubic $f = g_1g_2g_3$ consisting of lines $g_i \in \mathbb{C}[x,y]$. There is no rationality result in these cases. It is easy to see by way of the examples

$$f(x,y) = x^2 - 2y^2 = (x + \sqrt{2}y)(x - \sqrt{2}y)$$
$$f(x,y) = x^2 + 2y^2 = (x + \sqrt{2}iy)(x - \sqrt{2}iy)$$

that a line pair may involve two irrational lines that are real algebraic conjugates or complex conjugates of each other. A similar example for a cubic curve consisting of three algebraically conjugate lines forming a triangle is constructed in Section A.4.

By Proposition 2.3.4, a line $g = ay + bx + c$ is a component of f iff $\operatorname{res}(f,g,y) = 0$. Hence Proposition 2.4.3 implies that irrational lines come in algebraically conjugate groups.

For a triangle $f = g_1g_2g_3$ with vertices s_1, s_2, s_3 as in §3.3.2, there is an interesting link between the rationality of a line and the opposite vertex: If a line g_1 is rational, then $g_2g_3 = f/g_1$ and its unique singularity s_1 are also rational. Conversely, if s_1 is rational, we may translate such that $s_1 = 0$; then the quadratic part f_2 of f is just $f_2 = g_2g_3$ (since these are the tangents of f at s_1), proving that $g_1 = f/(g_2g_3)$ is rational, too.

3.3.4 Real Affine Phenomenology of Singularities

To understand the geometry of lines, conics, and cubics in the real affine plane, we need to refine their complex projective classification §3.3.2 by distinguishing real and complex components and tangents, and by excluding points at infinity.

For doing so, it is natural to consider curves $f \in \mathbb{R}[x,y]$. With Proposition 2.3.5 on complex conjugacy in place of Proposition 2.4.3 on algebraic conjugacy, our earlier arguments demonstrating that certain points (cf. §3.3.1) and components (cf. §3.3.3) are rational in the case of rational coefficients immediately translate to arguments demonstrating real points and real components in the case of real coefficients.

Lines and Conics

Let $f \in \mathbb{R}[x,y]$ be a line. Its zero set is indeed a line in the usual sense. The curve f has no singularities.

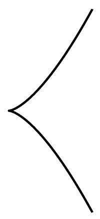
Let $f \in \mathbb{R}[x,y]$ be a conic. First assume that f is irreducible. Then it is either an ellipse, parabola, or hyperbola, or its zero set is empty (e. g. $f = x^2 + y^2 + 1$). Now assume that $f = g_1g_2$ is the product of two lines $g_1, g_2 \in \mathbb{C}[x,y]$. If the lines can be written as $g_1, g_2 \in \mathbb{R}[x,y]$, then the zero set consists of two lines in the usual sense. Unless these lines are parallel, they intersect in a singularity $v \in \mathbb{R}^2$ of multiplicity 2 with two real tangents (the lines themselves). Such a singularity is called a *crunode*. Otherwise, $V_{\mathbb{R}}(f) = g_1 \cap g_2$ consists of at most a single point v which is a singularity of multiplicity 2 with two complex-conjugate tangents (the two lines), called an *acnode*.

If f is y-regular, it is clear from §3.2.5 applied to the two components intersecting in a crunode that each branch can be parametrized analytically as $\gamma(x) = (x, \varphi(x))$ analogous to (3.19).

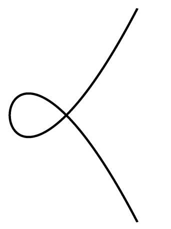
Irreducible Cubics

A cubic $f \in \mathbb{R}[x,y]$ can never have a finite (or even bounded) zero set, because after a suitable real affine change of coordinates, it is y-regular, such that substituting any $\xi \in \mathbb{R}$ for x yields a polynomial equation $f(\xi, y) = 0$ of degree 3 which has at least one solution in y. So a cubic is always visible as an infinite zero set in the real affine plane.

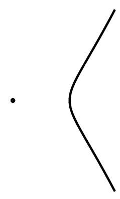
The unique singularity v of a singular irreducible real cubic f is real by Proposition 2.3.5 but may lie at infinity. If it does not lie at infinity, we can assume w. l. o. g. that $v = 0$ (otherwise translate). The two tangents at v (counted with multiplicities) are the linear factors of the quadratic part f_2 of f.



If $f_2 = l^2$ is a square, then v is a cusp with double tangent $l \in \mathbb{R}[x,y]$, and one can show [B98, Thm. 8.4] that a real projective change of coordinates yields Neil's parabola $f = y^2 - x^3$. It follows that there are two real branches of f converging to l on one side of v, which come out as two complex-conjugate branches on the other side. So the real picture shows two converging arcs on one side which do not continue on the other side.



If f_2 is not a square, it has two distinct linear factors, i. e. v is a node. As in the case of conics, v is either a crunode where two real branches of f intersect, or an acnode which is an isolated point of f, depending on whether the factors of f_2 are real or not. One can show [B98, Thm. 8.4] that a real projective change of coordinates transforms f to $f = y^2 - x^2(x+1)$ if it has a crunode or to $f = y^2 - x^2(x-1)$ if it has an acnode.



We want to show that the two branches in a crunode of an irreducible cubic are smooth, just like those at the intersection of two components. Using the technique of [G98, 8.1] one can find a rational parametrization

$$\begin{array}{rcl} \rho : \mathbb{R} & \to & \mathbb{R}^2 \\ t & \mapsto & (t^2 - 1,\, t(t^2 - 1)) \end{array} \tag{3.21}$$

for $f = y^2 - x^2(x+1)$. The crunode on f is $(0,0)$, and the continuously differentiable function ρ runs through $(0,0)$ exactly twice: For $t = -1$ with $\nabla\rho(t) = (-2, 2)$

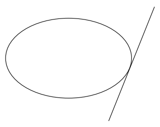
and for $t = +1$ with $\nabla\rho(t) = (2,2)$, corresponding to the two tangents $x \pm y$ of f at $(0,0)$. Hence ρ is a parametrization of both branches: around $t = -1$ for one and around $t = +1$ for the other. By inverting the change of coordinates used to obtain the normal form f, this extends to all irreducible cubics with a crunode. Let us restrict our considerations to non-vertical crunodes, so that $\rho_1'(\pm 1) \neq 0$.

To obtain a parametrization in the style of (3.19), observe that the rational functions ρ_1 and ρ_2 are in particular analytic functions. Since $\rho_1'(\pm 1) \neq 0$, the Inverse Function Theorem from complex analysis [L85, II.5] [R92, 9.4] guarantees the existence of an analytic function ρ_1^{-1} which inverts ρ_1 locally around $t = -1$ or $t = +1$, respectively. Thus, $\gamma(x) = (x,\ (\rho_2 \circ \rho_1^{-1})(x))$ is the desired smooth link.

Cubics with several components

Let us consider the case of a cubic $f \in \mathbb{R}[x,y]$ consisting of a line g and an irreducible conic h with equations $g, h \in \mathbb{C}[x,y]$. In complete analogy to the argument in §3.3.3, we obtain that $g, h \in \mathbb{R}[x,y]$.

If g is a chord of h, an intersection of g and h in $\mathbb{R}^2$ is a crunode, the respective tangents being g itself and the tangent to h at the point. If g is a tangent to h, then the unique singularity v of f is real but may lie at infinity. Assume $v \in \mathbb{R}^2$. The geometry of f around v is different from what we called a cusp for irreducible real cubics: There are two real branches sharing the same tangent g, but both of them exist in the real plane on both sides of v along the common tangent. Therefore, a real singularity of this kind is given a different name and commonly called a *tacnode*.



Let us now consider the case that $f = g_1 g_2 g_3$ factors into three lines. If these lines form a star, there is a unique and hence real singularity v with three distinct tangents, viz. g_1, g_2, and g_3. In analogy to the conjugation argument in §3.3.3, we see that exactly one or three of them are real. Depending on this, we call v a *complex* or *real triple point*, respectively.

If f is a triangle, then exactly one or three of the vertices s_1, s_2, and s_3 are real. If all three are real, all lines have to be real, and each vertex $s_i \in \mathbb{R}^2$ is a crunode. Conversely, if only s_1 is real, then it is an acnode, and just one of the lines, viz. $g_1 \not\ni s_1$, is real.

By the same argument used for a conic with a crunode, we obtain parametrizations analogous to (3.19) for all real branches of the singularities just discussed.

Arcs and Singularities

Let us now summarize how these different *kinds* of singularities link the arcs of f. Assume that coordinates are chosen such that the singularity v is not vertical. Then all arcs converging to v, even arcs on line components, can be said to come from one side of the horizontal line $x = v_1$ and tend to the other side.

- An **acnode** is simply an isolated point and does not link any arcs.
- A **crunode** has two real branches that change sides. Each branch forms a smooth link between two arcs, one on either side, so that altogether four arcs meet at v.
- A **tacnode** has two real branches that do *not* change side. Each branch forms a smooth link between two arcs, one on either side, so that altogether four arcs meet at v.
- A **cusp** has two arcs coming in on one side, so that we can speak of a *left* or *right cusp* in analogy to left and right x-extreme points. Both arcs are topologically connected, but they are not linked smoothly.
- A **real triple point** v has three real branches, any two of which change sides at v. Each branch forms a smooth link between two arcs, one on either side, so that altogether six arcs meet at v.
- A **complex triple point** has one real branch with a zero set that looks like a non-singular piece of curve. The two complex branches intersecting at that point are not visible in the real zero set.

The term *smooth link* of two arcs A_1 and A_2 means that there is a branch of f at the singularity (r,s) which can be parametrized analytically in the style of (3.19) as $\gamma :]r-\varepsilon, r+\varepsilon[\to \mathbb{R}^2$, $x \mapsto (x, \varphi(x))$ such that $\gamma(r) = (r,s)$, that $\gamma|_{]r-\varepsilon,r]} \subseteq A_1$, and that $\gamma|_{[r,r+\varepsilon[} \subseteq A_2$.

For algebraic curves of arbitrary degrees, the problem of tracing branches through singularities is addressed by the theory of Puiseux series (see [BK86]) which is much more general.

3.4 Parametrized Curves

The object of this section is to prove general statements which allow us to conclude that cubic splines can be turned into (segments of) cubic curves. This may sound tautological, but recall that the most general form of a cubic spline, including *non-uniform rational B-splines* (NURBS), is a parametrized curve

$$\begin{aligned} \gamma : D &\to K^2 \\ t &\mapsto \left(\frac{a(t)}{c(t)}, \frac{b(t)}{c(t)}\right) \quad \text{with } a,b,c \in K[t] \end{aligned} \tag{3.22}$$

where $K = \mathbb{R}$, $D = [l,r]$, and "cubic" refers to the degree ≤ 3 of the polynomials a, b, c in the parametrization. This includes the case $c = 1$ for non-rational splines, and it includes the parametrization of a singular cubic that we met in §3.3.4. We will be lazy in the sequel and write just γ for the point set $\gamma(D)$.

Let us address the task of making a parametrized curve (3.22) implicit in the case where K is algebraically closed and γ has the maximal domain $D = K \setminus \mathrm{V}(c)$. A point (ξ,η) lies on γ iff there is a parameter $\tau \in D$ so that $a(\tau)/c(\tau) = \xi$ and $b(\tau)/c(\tau) = \eta$. We can view this as an intersection of $P(x,y,t) = a(t) - xc(t)$ and $Q(x,y,t) = b(t) - yc(t)$ in K^3. We can always choose a,b,c such that $\gcd(a,b,c) = 1$, and we assume this in the sequel. Then all points $(\xi,\eta,\tau) \in P \cap Q$ satisfy $c(\tau) \neq 0$. It is clear that all points of γ lie on the algebraic curve $f := \mathrm{res}(P,Q,t)$. It remains to show that f is not "too big". The following two proposition are relevant.

Proposition 3.4.1: *Let γ be as above with $n = \max\{\deg(a),\deg(b),\deg(c)\} \geq 1$. Then the total degree of $f := \mathrm{res}(a(t) - xc(t),\ b(t) - yc(t),\ t)$ is $\deg(f) \leq n$.*

Proof: The proof is a sequence of matrix transformations, and we depict it for the case $n = 3$. The extension to other values of n is straightforward. Let a_i denote the coefficient of t^i in the polynomial a etc., and consider the following matrix:

$$S = \begin{pmatrix} a_3 - c_3x & a_2 - c_2x & a_1 - c_1x & a_0 - c_0x & & \\ & a_3 - c_3x & a_2 - c_2x & a_1 - c_1x & a_0 - c_0x & \\ & & a_3 - c_3x & a_2 - c_2x & a_1 - c_1x & a_0 - c_0x \\ b_3 - c_3y & b_2 - c_2y & b_1 - c_1y & b_0 - c_0y & & \\ & b_3 - c_3y & b_2 - c_2y & b_1 - c_1y & b_0 - c_0y & \\ & & b_3 - c_3y & b_2 - c_2y & b_1 - c_1y & b_0 - c_0y \end{pmatrix}$$

This matrix S is the Sylvester matrix $\mathrm{Syl}(P,Q,t)$, provided that $\deg(P) = \deg(Q) = n$. This does hold if $\deg(c) = n$. If $\deg(c) < n$, then at least one of $\deg(a)$, $\deg(b)$ is n, so that $\det(S)$ is a scalar multiple of $f = \det(\mathrm{Syl}(P,Q,t))$ by the argument used in the proof of Proposition 2.3.3. In any case, it suffices to argue about the degree of $\det(S)$.

Let $d := \deg(c)$ and $k = n - d + 1$. Then $c_d \neq 0$, and the k-th column of S is the first to contain x and y in the first and $(n+1)$-st row. Subtracting c_{d+k-i}/c_d times the k-th column from the i-th column for $i = k+1,\dots,k+d$ eliminates all occurrences of x and y in the first and $(n+1)$-st row except in column k and does not affect the coefficients of x and y in the other rows. The same process repeated for $k = n - d + 2,\dots,2n - d$ eliminates all but the respective first occurrences of x and y in the remaining rows, leaving us with a matrix containing x and y only in the n columns $n - d + 1$ to $2n - d$ (depicted for $n = d = 3$):

$$S' = \begin{pmatrix} \dots x \dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ & \dots x \dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ & & \dots x \dots & \dots\dots & \dots\dots & \dots\dots \\ \dots y \dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ & \dots y \dots & \dots\dots & \dots\dots & \dots\dots & \dots\dots \\ & & \dots y \dots & \dots\dots & \dots\dots & \dots\dots \end{pmatrix}$$

The determinant $\det(S') = \det(S)$ of this matrix has total degree n in x,y because each of its terms contains exactly one entry from each column. $\square$

We will only argue heuristically[3] that a lower degree than n cannot be attained in general: With non-degenerate choices of γ and an arbitrary line $g = ux + vy + w$, we expect that the polynomial $h(t) = ua(t) + vb(t) + wc(t)$ has degree n and has n distinct zeroes, none of which lies in $K \setminus D$. This means that γ intersects g in n distinct points, hence any algebraic curve $f \supseteq \gamma$ has to have degree $\geq n$ by Bezout's Theorem. However, specific choices of γ may lead to curves that are not square-free: Consider the example $a(t) = t^3 - t$, $b(t) = a(t) - 1$, $c(t) = 1$. Here $\gamma : t \mapsto (a(t)/c(t),\ b(t)/c(t))$ parametrizes the line $x - y - 1$ non-injectively, and implicitizing it as above yields $f = (x - y - 1)^3$.

Once we forget about multiplicity of components and focus on the points sets f and γ, we see that the algebraic curve f is chosen optimally:

Proposition 3.4.2: *Let K be algebraically closed, and let f and γ be as in Proposition 3.4.1. Let g be the square-free part of f. Then g is the unique square-free algebraic curve such that $\gamma \subseteq g$, and that $g \setminus \gamma$ is finite. The polynomial g is irreducible in $K[x,y]$.*

Proof: We first demonstrate the properties of g. The inclusion is clear. Let (ξ,η) be a point on g. Then $f(\xi,\eta) = 0$. The argument employed earlier in the proof of Proposition 2.3.3 shows that whenever the first column of S does not vanish, then $f(\xi,\eta) = 0$ implies there is a point $(\xi,\eta,\tau) \in P \cap Q$, meaning that $(\xi,\eta) = \gamma(\tau)$. The vanishing of the first column of S is equivalent to $a_n - c_n\xi = b_n - c_n\eta = 0$. This can only hold if $c_n \neq 0$, and then only at the one point $v_\infty = (a_n/c_n,\ b_n/c_n)$.

Hence we have $f = \gamma$ for $c_n = 0$ and $f = \gamma \cup \{v_\infty\}$ for $c_n \neq 0$, proving the finite complement property of f. (Note that v_∞ may or may not already be included in γ.)

To see that these properties of g imply irreducibility, assume $g = g_1 g_2$ had two proper coprime factors g_1, g_2. Since $g(\gamma(t))$ vanishes for all of the infinitely many $t \in K$, one of the factors, say g_1, also has the property that $g_1(\gamma(t)) = 0$ for infinitely many $t \in K$. Hence this equality holds for all $t \in K$, meaning that $\gamma \subseteq g_1$. The desired contradiction comes from the observation that g_2 contains infinitely many points, but only a finite number of them, viz. the intersections with g_1 (cf. Bezout's Theorem), lie on γ; disproving the finite complement property of $g \supseteq g_2$.

Now it is easy to see uniqueness: Assume h is an algebraic curve with $\gamma \subseteq h$ and $h \setminus \gamma$ finite. Then the preceding arguments show that h, like g, is irreducible. Since it has the infinitely many points of γ in common with g, the two must be equal (as curves, i. e. up to constant factors). $\square$

The point v_∞ appearing above has an obvious meaning for $K = \mathbb{C}$: It is the limit of $\gamma(t)$ for $t \to \infty$. One can see topologically that it has to be contained in g: Every algebraic curve is a closed subset of $\mathbb{C}^2$ (since it is the preimage of $\{0\}$), and thus has to contain a limit like this one.

[3] The author regrets that he could not follow a hint kindly provided by Frank Schreyer stating that rigorous reasoning based on the same idea is possible.

As the special role of the point v_∞ and the divergence of $\gamma(t)$ for t approaching a zero of $c(t)$ suggests, a projective setting is more natural for parametrized curves, but this is beyond the scope of this text.

Let us return to our original aim of implicitizing splines. Proposition 3.4.1 allows us to make a cubic spline γ implicit and obtain a cubic curve f containing it. Proposition 3.4.2 shows that f is the optimal choice of an algebraic curve supporting γ, unless γ is a line. Our algorithm will support the notion of a segment of a curve to reflect that γ is a bounded subset of f due to its bounded parameter interval.

The parametrization of a spline curve certainly allows other approaches to intersection computation than the one developed in this text, which relies exclusively on equations defining curves implicitly. However, the focus on implicit curves is justified by the fact that "most" cubic curves do not allow a rational parametrization like (3.22): A non-singular curve of degree ≥ 3 cannot have a rational parametrization [G98, Lemma 18.8]. (For $d = 3$, the converse also holds: A rational parametrization is always possible for singular cubic curves [G98, 8.1]; we have already met one example in §3.3.4.)

Chapter 4

Algorithmic Analysis of Cubic Curves and Curve Pairs

This chapter presents a practical algorithm for the geometric analyses of algebraic curves of degree up to 3 and of pairs of them. These analyses provide the basis for the arrangement computation algorithm of Chapter 5.

4.1 Analyzing One Curve

4.1.1 Basics

The general view we take on the geometry of a real algebraic curve $f \in \mathbb{Q}[x,y]$ is as follows: For a given x-coordinate $\xi \in \mathbb{R}$, how many real points of f exist *over* ξ (that is with an x-coordinate equal to ξ), and how does this change as we vary ξ? In the style of the sweep line algorithm for arrangement computation, this question can be rephrased as: How does the intersection of f with a vertical line $g = x - \xi$ evolve as we sweep g from $-\infty$ to $+\infty$?

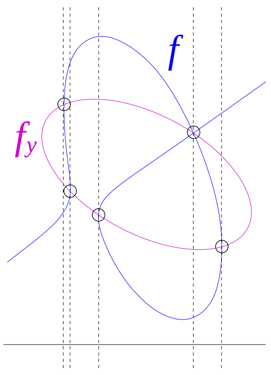
The general structure of the answer was given in §3.2.5, based on the Implicit Function Theorem: In the neighbourhood of all but finitely many ξ, the arcs of f evolve smoothly, according to their parametrizations (3.19), and do not touch each other. There is only a finite set $f \cap f_y \subseteq \mathbb{R}^2$ of points at whose x-coordinates ξ "something happens", i. e. at which the number of points over ξ changes or where two arcs have a point in common. Therefore, we call the points of $f \cap f_y$ the *(one-curve) event points* of f, and their x-coordinates are called the *(one-curve) event x-coordinates*.

The rest of this section describes an algorithm that takes an algebraic curve $f \in \mathbb{Q}[x,y]$ with degree $\deg(f) \leq 3$, subject to certain conditions, and determines:

- The decomposition of the x-axis into event x-coordinates and open intervals between them.
- The number of arcs over any $\xi \in \mathbb{R}$.
- The *kind* of each event, i. e. left/right x-extreme point or kind of singularity as discussed in §3.3.4.
- The arcs involved in each event, and their position relative to the other arcs.

Since a line has exactly one arc and no events, we restrict our presentation to the cases of conics and cubics.

The general idea upon which the algorithm rests is to regard locating the events as an intersection problem involving f and f_y. Partial solutions of the system $f = f_y = 0$, that is x-coordinates of event points, are determined by finding the real zeroes of $\mathrm{res}(f, f_y, y)$. However, we want to avoid the costly arithmetic involved in computing the symbolic representation of the matching y-coordinates, where possible. Instead, we stick to the "y per x" point of view and often determine the y-coordinate of a point just implicitly by identifying the arc of f containing it. Some central ideas of how to do this have appeared before in Wolpert's thesis [W02]. In the same fashion, we do not care about the actual parametrizations of arcs, just their relative position. Therefore, our analysis could be described as mostly topological. Computing numerical coordinate data can take place as a post-processing step (cf. §4.3.3).



We employ the method of §2.4.4 to represent real zeroes of rational polynomials, in particular event x-coordinates, by a suitable factor of the polynomial and an isolating interval. This allows us to perform three-valued comparisons (less-equal-greater) for any two such numbers (see ibid.).

Preferring rational over algebraic arithmetic has strongly influenced the design of the algorithm, and we formulate it in terms of rational arithmetic. An implementation should go further and use integer instead of rational arithmetic where possible. The consequences of Gauss' Lemma presented in §2.2.2 pave the way for this. If f has integer coefficients, then many polynomials appearing during the algorithm can share this property, in particular resultants and gcds.

The following conditions are imposed on the curve f, besides the degree bound:

- The curve f is square free.
- The curve f is y-regular (*leading coefficient condition*).
- No two points of $V_{\mathbb{C}}(f) \cap V_{\mathbb{C}}(f_y)$ are covertical.
- There are no vertical flexes on f.
- There are no vertical singularities on f.

Two points (a,b), (a',b') are *covertical* if $a = a' \wedge b \neq b'$. Recall that a point has been called *vertical* in §3.2.4 if it has a vertical tangent.

These conditions are checked by the algorithm as far as it needs to rely on them. (The noncoverticality condition holds automatically by Proposition 3.3.1.) Violations are signalled to the caller and cause the algorithm to abort. It is the caller's responsibility to establish the conditions and to restart the algorithm if necessary. For a y-regular curve f, squarefreeness can be obtained by replacing f with $f/\gcd(f, f_y)$, see §2.2.3. For the other conditions, see Section 4.3.

The violation of a condition does not cause an incorrect result. Also, these conditions do not limit the range of permissible input curves (seen as point sets), just the choice of an equation (squarefreeness condition) or of a coordinate system (other conditions) to represent them. Since the conditions can be established mechanically, as discussed in Section 4.3, the resulting algorithm remains complete.

4.1.2 Event x-coordinates

The algorithm begins with a resultant computation. Recall that f is square free iff $\operatorname{res}(f, f_y, y) \neq 0$, see §2.2.3, §2.3.2.

> **Resultant:** If the coefficient of $y^{\deg(f)}$ in f is zero, abort.
> Compute $R_f := \operatorname{res}(f, f_y, y)$. If $R_f = 0$, signal "not square free" and abort.

Let $x_1 < x_2 < \ldots < x_n$ be the distinct real zeroes of R_f, and let x_i be one of them. By Proposition 2.3.3 there exists at least one point of $\mathrm{V}_{\mathbb{C}}(f) \cap \mathrm{V}_{\mathbb{C}}(f_y)$ with an x-coordinate equal to x_i. By Proposition 3.3.1 there exists at most one point of $\mathrm{V}_{\mathbb{C}}(f) \cap \mathrm{V}_{\mathbb{C}}(f_y)$ with an x-coordinate equal to x_i, and its y-coordinate is also real. Hence we have a bijection between the $n \geq 0$ real zeroes of R_f (the event x-coordinates) and the n points of $f \cap f_y$.

The event x-coordinates induce a partition of the x-axis:

$$\mathbb{R} \ = \ ]-\infty, x_1[\cup \{x_1\} \cup]x_1, x_2[\cup \{x_2\} \ \cup \ \ldots \ \cup \ \{x_n\} \cup]x_n, +\infty[. \tag{4.1}$$

Call $]x_{i-1}, x_i[$ the *i-th interval between events*. For simplicity, let $x_0 = -\infty$, $x_{n+1} = +\infty$, and use the word "between" also for the first ($i = 1$) and last ($i = n+1$) interval between events.

The multiplicity m_i of an event x-coordinate x_i as a zero of R_f distinguishes kinds of events: For an x-extreme point we have $m_i = 1$ (see §3.2.5), whereas for a singularity we have $m_i \geq 2$ (by Proposition 3.2.6).

> **One-curve Event x-Coordinates:**
> Factor R_f by multiplicities and obtain $R_f = \prod_{m=1}^{M} R_{fm}^{m}$.

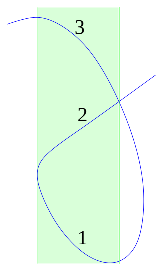
Represent the real zeroes of each non-constant square-free factor R_{fm} as pairs of defining polynomial and isolating interval (§2.4.4).

Merge the real zeroes of all factors to yield the sorted sequence of event x-coordinates $x_1 < x_2 < \ldots < x_n$ and record their respective multiplicities m_i as zeroes of R_f.

Invoke the method of §4.1.3 to check the absence of vertical flexes.

We have to compute information on the *behaviour* of f, that is on the arcs and events of f, over the event x-coordinates and over the intervals between them. The latter is easy:

The set $B = f \cap (I \times \mathbb{R})$ of real zeroes of f over the i-th interval I between events consists of $k_i \leq \deg(f)$ connected components $B_1, \ldots, B_{k_i}$, each of which equals $B_\mu = \text{interior}(A_\mu) \cap (I \times \mathbb{R})$ for some arc A_μ of f. Since arcs do not cross between events, we may sort the B_μ in ascending y-order. That is, we may choose indices $1, \ldots, k_i$ such that for all $x \in I$, $(x, y_\mu) \in B_\mu$, and $(x, y_\nu) \in B_\nu$ it holds that $\mu \leq \nu \Leftrightarrow y_\mu \leq y_\nu$. Referring to this order, we say that the arc A_μ has the *arc number μ on f over I*, or is *the μ-th arc of f over I.*



Arcs over Intervals: Using the rational isolating intervals of the x_i, determine a rational point $r_i \in]x_{i-1}, x_i[\cap \mathbb{Q}$ within each interval between events.

Count the real zeroes of $f(r_i, y) \in \mathbb{Q}[y]$ to determine the number k_i of arcs of f over the i-th interval between events.

The y-regularity condition in conjunction with Corollary 3.2.10 implies that the polynomial $f(\xi, y) \in \mathbb{R}[y]$ has degree $\deg(f)$ and is square free whenever ξ is not an event x-coordinate. Hence it follows from Proposition 2.3.5 that $k_i \in \{0, 2\}$ for $\deg(f) = 2$ and $k_i \in \{1, 3\}$ for $\deg(f) = 3$.

We will start to investigate the behaviour of f at an event point in §4.1.4. But first we deal with vertical flexes.

4.1.3 Excluding Vertical Flexes

Our task is to determine whether f has vertical flexes, and if so, signal this situation and abort the algorithm. Recall that a point $(r, s) \in \mathbb{R}^2$ is a vertical flex of f iff $f(r, s) = f_y(r, s) = f_{yy}(r, s) = 0$ and $f_x(r, s) \neq 0$. Also recall that flexes do not exist for $\deg(f) \leq 2$.

We know from Proposition 3.3.3 that vertical flexes correspond to zeroes of R_{f2}. Each real zero x_i of R_{f2}, also called a *suspect*, corresponds to an event point $(x_i, y_i) \in f \cap f_y$. For all suspects x_i we have to check that $(x_i, y_i) \notin f_{yy} \setminus f_x$.

To keep algebraic degrees down, we begin with inspecting the intersection $f_y \cap f_{yy}$ of a conic and a line. Because f is y-regular, so are f_y and f_{yy}.

If $R := \mathrm{res}(f_y, f_{yy}, y) = 0$, then the zero set of $f_y = \frac{1}{2} f_{yy}^2$ is just the line f_{yy}, so that a point $(x_i, y_i) \in f \cap f_y$ fulfills the desired condition $(x_i, y_i) \notin f_{yy} \setminus f_x$ iff it holds that $(x_i, y_i) \in f_x \cap f_{yy}$. Since f_{yy} is a line, its points correspond bijectively to their x-coordinates, so we can check just as well that all suspects x_i are zeroes of $Q := \mathrm{res}(f_x, f_{yy}, y)$. To this end, consider $d := \gcd(R_{f2}, Q)$ which is square free and has degree ≤ 2. Hence the number r of its real zeroes is easily determined by inspecting degree and, if not linear, its discriminant. If r is less than the number of suspects, some suspect corresponds to a vertical flex. Otherwise, every suspect x_i comes from a point $(x_i, y_i) \in f_x \cap f_{yy}$ which is not a vertical flex.

Vertical Flexes, part 1: Call all real roots of R_{f2} suspects.

If $\deg(f) \leq 2$ or if there are no suspects, f has no vertical flexes.

Compute $R := \mathrm{res}(f_y, f_{yy}, y)$.

If $R = 0$, count the r real zeroes of $\gcd(R_{f2}, \mathrm{res}(f_x, f_{yy}, y))$.
f has no vertical flexes iff r equals the number of suspects.

Now we may assume $R \neq 0$. Consider a suspect x_i. If $R(x_i) \neq 0$, then (x_i, y_i) is clearly not a vertical flex, since f_y and f_{yy} do not intersect over x_i.

If $R(x_i) = 0$, then $f_y(x_i, y) \in \mathbb{R}[y]$ has a double root y_0 by Proposition 2.2.10, meaning that $f_y(x_i, y) = \ell(f_y)(y - y_0)^2$ and $f_{yy}(x_i, y) = \ell(f_{yy})(y - y_0)$. Since $R_f(x_i) = 0$, it follows further by the same reasoning that $f(x_i, y) = \ell(f)(y - y_0)^3$. Hence the unique point of f over x_i is (x_i, y_i), and its y-coordinate $y_i = y_0$ is the solution of the linear equation $f_{yy}(x_i, y) = 0$. This point (x_i, y_i) is a flex iff $f_x(x_i, y_i) \neq 0$

The suspects x_i for which R vanishes are precisely the zeroes of the square-free polynomial $h := \gcd(R, R_{f2})$ which has degree at most 2. Hence for $\deg(h) > 1$ the evaluation of $f_x(x_i, y_i)$ can either be implemented by computation in $\mathbb{Q}[x]/(h)$ (§2.4.2) or by arithmetic in $\mathbb{Q}(\sqrt{D})$, $D > 0$ using separation bounds (§2.4.3).

Vertical Flexes, part 2: Forget all suspects x_i with $R(x_i) \neq 0$.

For all remaining suspects x_i, compute y_i as the unique solution of the linear equation $f_{yy}(x_i, y) = 0$. If $f_x(x_i, y_i) \neq 0$, there is a vertical flex.

If no vertical flexes have been found by now, f has none.

This solves the task of checking for vertical flexes.

4.1.4 Arcs over event points

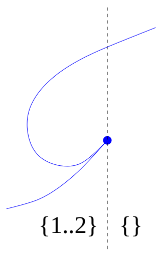
The rest of this section is concerned with the analysis of f at event points (x_i, y_i). First, this requires to find out about the k_i' distinct real points of f over x_i, i. e.

the zeroes of $f(x_i,y) \in \mathbb{R}[y]$, counted *without* multiplicities. To unify terminology with the analysis over intervals between events, we say that a point (x_i,y) has *arc number* μ, $1 \le \mu \le k_i'$, over x_i if it is the μ-th point of f over x_i, numbered in ascending order of y-coordinates, without multiplicities.

We know $k_i' \ge 1$, since there is at least the event point itself. On the other hand, $k_i' \le \deg(f) - 1$ by Corollary 3.2.10. Hence for $\deg(f) = 2$ we know $k_i' = 1$.

For $\deg(f) = 3$, we have to determine whether $k_i' = 1$ or $k_i' = 2$. In the latter case, the question arises if the other zero $y_i' \ne y_i$ of $f(x_i,y)$ is smaller or larger than y_i.

So much for $f \cap \{x_i\} \times \mathbb{R}$. Let us now take an infinitesimally broader view and consider the neighbourhood of x_i. The question arises how the arcs of f over the *incident intervals* on the *left-hand side* $]x_{i-1},x_i[$ and on the *right-hand side* $]x_i,x_{i+1}[$ connect with the points of f over x_i. A non-event point $(x_i,y_i') \ne (x_i,y_i)$ lies in the interior of exactly one arc of f (called a *continuing* arc) which extends to both sides of the event. The arcs of f containing the event point (x_i,y_i) are said to be *involved* in this event, and (x_i,y_i) is a boundary point of each of them.



Consider one of the intervals I incident to the event x-coordinate. If two arcs with arc numbers $i < j$ over I are involved, then so are all arcs with arc numbers $i, i+1, \dots, j$. Hence the range of involved arc numbers over I is either empty or can be represented by a pair of minimum and maximum arc number.

In the sequel, we describe how to compute the following information for each event point (x_i,y_i):

- The kind of event (left/right x-extreme or kind of singularity from §3.3.4).
- The number k_i' of distinct points on f over x_i.
- The arc number of the event point over x_i.
- The range of arc numbers of the arcs involved in the event on either side.

The kind of event and $\deg(f)$ imply the value of k_i' and the number of involved arcs on either side. If $k_i' > 1$, then it remains to sort the points over x_i. Since $\deg(f) \le 3$, there is at most one such point (x_i,y_i'), and comparing y_i to y_i' also determines the arc numbers of the one continuing arc and the involved arcs (if any) on either side.

The analysis of an event is split into three parts. If $m_i = 1$, use the method of §4.1.5. For $m_i > 1$, use the methods of §4.1.6 or §4.1.7, depending on $\sum_{m \ge 2} \deg(R_{fm})$, i. e. the number of singular points of f in $\mathbb{C}^2$.

4.1.5 Finding x-extreme points

A zero x_i of R_f with multiplicity $m_i = 1$ corresponds to an x-extreme point (x_i,y_i) of f. We have to determine whether it is a left or right x-extreme point and which arcs it involves.

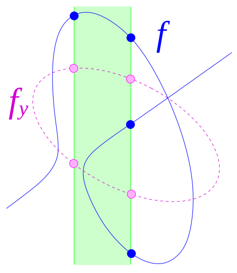
The point (x_i, y_i) is a left or right x-extreme point if $k_{i+1} - k_i$ equals $+2$ or -2, respectively. For simplicity, let us assume that (x_i, y_i) is a left x-extreme point. The other case is symmetric.

For $\deg(f) = 2$, there are no arcs to the left of this event, and the event involves both arcs of f, numbered 1 and 2, to its right; completing the analysis in this case.

For $\deg(f) = 3$, one arc of f continues from the left (where it has the number 1) to the right, and two new arcs begin and extend to the right. It remains to find out whether they lie below or above the continuing arc.

By the definition of an x-extreme point, $f_y(x_i, y_i) = 0$ and $f_{yy}(x_i, y_i) \neq 0$. Hence the quadratic polynomial $f_y(x_i, y)$ has two simple roots, so that f_y and f_{yy} do not intersect over x_i. Therefore we have $\operatorname{res}(f_y, f_{yy}, y)(x_i) \neq 0$, and we can refine the isolating interval $]r_i, r_{i+1}[$ of x_i to an isolating interval $]r_-, r_+[$ of x_i with rational boundaries so that $I = [r_-, r_+]$ contains no zero of $\operatorname{res}(f_y, f_{yy}, y)$, cf. §2.4.4. This means that the curve f_y has no event points over I.

For all $\xi \in I$, the Mean Value Theorem from calculus implies that between any two zeroes of $f(\xi, y) \in \mathbb{R}[y]$ there is a zero of $f_y(\xi, y) \in \mathbb{R}[y]$, so that the arcs of f_y, which extend over I, separate the arcs of f. In particular, when we compute the sequence of intersections of f and f_y with the horizontal lines $x - r_-$ and $x - r_+$, sorted in ascending order of y-coordinates, we can match the two intersections of f_y on either line and find out whether the continuing arc of f has arc number 1 or arc number 3 on the right-hand side of the event.



This method of extending an isolating interval $I \ni x_i$ to a stripe $I \times \mathbb{R} \subseteq \mathbb{R}^2$ and inferring information about an event inside the stripe from intersections of curves with the vertical lines bounding the stripe is our main method of avoiding explicit arithmetic with an event x-coordinate x_i. It can be seen as a variant of box hit counting [W02] [KC$^+$99].

> **x-Extreme Point:** Determine $\operatorname{sign}(k_{i+1} - k_i)$ to distinguish between a left or right x-extreme point.
>
> If $\deg(f) = 2$, this distinction describes the event point completely.
> The rest applies only to $\deg(f) = 3$.
>
> Take the interval $]r_i, r_{i+1}[$ and refine it to an interval $]r_-, r_+[\ni x_i$ so that $[r_-, r_+]$ does not contain any zero of $\operatorname{res}(f_y, f_{yy}, y)$.
>
> Let $r = r_-$ for a left and $r = r_+$ for a right x-extreme point.
>
> If the unique real zero of $f(r, y) \in \mathbb{Q}[y]$ is larger than the two real zeroes of $f_y(r, y) \in \mathbb{Q}[y]$, the two arcs involved have numbers 1 and 2, otherwise numbers 2 and 3.

The last step can be performed by comparing the signs of $\ell(f)$ (which is the sign of $f(r,y)$ for $y \to +\infty$) and the sign of $f(r,y)$ at the zero of $f_{yy}(r,y)$ (which lies between the two zeroes of $f_y(r,y)$): Iff they are equal, the unique real zero of $f(r,y)$ is smaller than the two real zeroes of $f_y(r,y)$.

4.1.6 Unique Singularities

Our task is to analyze a singularity (x_i,y_i) of f which we know to be the only singularity of f in $\mathbb{C}^2$. As we go along, we have to check the requirement that f does not have a vertical tangent in (x_i,y_i).

The coordinates (x_i,y_i) are rational: One can obtain $x_i \in \mathbb{Q}$ immediately from the resultant factor $R_{fm_i} = x - x_i \in \mathbb{Q}[x]$, because the uniqueness of the singularity is equivalent to $\sum_{m\geq 2}\deg(R_{fm}) = 1$. Next $y_i \in \mathbb{Q}$ can be obtained by factoring $f(x_i,y) \in \mathbb{Q}[y]$ by multiplicities, since Corollary 3.2.10 implies that y_i is the unique multiple root.

To follow the case distinction of Section 3.3, we translate f to $\hat{f}(x,y) = f(x+x_i,y+y_i)$. Write $\hat{f} = \hat{f}_3 + \hat{f}_2$ as sum of its homogeneous parts (see (3.7)). Constant and linear part vanish since $\mathrm{mult}((0,0);\hat{f}) \geq 2$. Either $\hat{f}_3$ or $\hat{f}_2$ might also be zero.

Two observations allow us to take shortcuts in computing $\hat{f}$: Let $d := \deg(f) = \deg(\hat{f})$. The highest-order terms f_d of f are invariant under translation, so that $\hat{f}_d = f_d$. For $d = 3$, the quadratic part $\hat{f}_2$ can be computed by evaluating partial derivatives as in (3.13).

For a conic, the kind of singularity (crunode or acnode) is all that needs to be determined. For a singularity of multiplicity 2 on a cubic, we have to find out its kind (crunode, cusp, acnode, or tacnode) and to determine which arcs of f are involved. For a singularity of multiplicity 3 on a cubic, we have to tell apart the two kinds of triple points (complex or real). The main distinguishing feature of the kinds of singularities is the number and multiplicity of real tangents, which correspond to the real roots of the LOT of $\hat{f}$.

The nonverticality requirement only has to be checked in the case of a cubic with a singularity of multiplicity 2. In the other cases, the tangents at (x_i,y_i) are components of f. Thus, they are y-regular by Proposition 2.2.1 and hence not vertical.

> **Unique Singularity:** Let $(x_i,y_i) \in \mathbb{Q}^2$ be the unique singularity of f in $\mathbb{C}^2$.
> Find unique zero x_i of R_{fm_i}. Find unique multiple zero y_i of $f(x_i,y) \in \mathbb{Q}[y]$.
> If $\deg(f) = 2$, then $\hat{f} = f_2 = ay^2 + bxy + cx^2$. Let $\sigma = \mathrm{sign}(b^2 - 4ac)$.
> If $\sigma > 0$, then we have a crunode, involving both arcs on both sides.
> Else $\sigma < 0$, we have an acnode, and there are no arcs on either side.
> The rest applies only to $\deg(f) = 3$.
> Compute $\hat{f}(x,y) = \hat{f}_3(x,y) + \hat{f}_2(x,y) = f_3(x,y) + ay^2 + bxy + cx^2$.

If $\hat{f}_2 = 0$, we have a triple point:
It is a real triple for $k_i = 3$ and a complex triple for $k_i = 1$.
In either case, it involves all $k_i = k_{i+1}$ arcs on both sides.

If $\hat{f}_2 \neq 0$, let $\sigma = \operatorname{sign}(b^2 - 4ac)$.
For $a = 0$, signal the error "vertical singularity" and abort.
For $\sigma > 0$, we have a crunode.
For $\sigma < 0$, we have an acnode.
For $\sigma = 0$ and $|k_{i+1} - k_i| = 2$, we have a cusp.
For $\sigma = 0$ and $|k_{i+1} - k_i| = 0$, we have a tacnode.
Factor $\hat{f}(0,y) = \ell(\hat{f})(y - y_0)y^2$ by setting $y_0 = -a/\ell(f)$.
If $y_0 > 0$, the continuing arc runs above the singularity, else it runs below.

Cusps and tacnodes can also be distinguished without using arc counts: If $\sigma = 0$, then $\hat{f}_2 \sim g^2$ with the double tangent $g = vx - uy$ at $(0,0)$. A tacnode appears iff f has two components, conic and chord, which is the case iff $g | \hat{f}_3$, which is in turn equivalent to $f_3(u,v) = 0$.

Crunode and acnode on a conic can also be distinguished by testing whether $k_i = k_{i+1} = 0$ or $k_i = k_{i+1} = 2$. Conversely, real and complex triple points can also be distinguished by the discriminant of $\hat{f}_3$ whose sign indicates whether there are one or three real tangents [BK86, pp. 182+].

Since analyzing singularities is not at all performance critical for arrangement computation, an efficiency comparison of these alternatives has not been conducted.

4.1.7 Multiple Singularities

Let f have exactly $s = \sum_{m \geq 2} \deg(R_{fm}) > 1$ singular points in $\mathbb{C}^2$, and let exactly r of them be elements of $\mathbb{R}^2$. For $r = 0$ there is nothing to be done. So let $(x_i, y_i) \in \mathbb{R}^2$ be one of these singular points.

Our task is to analyze the real singularity (x_i, y_i), based on the case distinctions of Section 3.3. They imply that $\deg(f) = 3$ and that (x_i, y_i) is a node, resulting from the intersection of two components. By Corollary 3.2.10 we know $f(x_i, y) = \ell(f)(y - y_i')(y - y_i)^2$ for some $y_i' \in \mathbb{R}$. The required condition that f shall not have a vertical tangent in (x_i, y_i) is equivalent to $y_i \neq y_i'$, because $y_i = y_i'$ implies tangency of the vertical line $x - x_i$ by Propositions 3.2.6 and 3.2.3. Hence there has to be a non-event point (x_i, y_i') on a continuing arc of f over x_i.

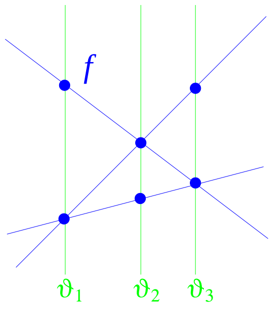
What we have to find out is:

- Does the nonverticality condition hold?
- Does the continuing arc of f run above or below (x_i, y_i)?
- Is (x_i, y_i) a crunode or an acnode?

Let us begin with the second question. It is equivalent to computing $\operatorname{sign}(y_i - y_i')$. By Proposition 3.3.1 the algebraic degree of y_i is no larger than $\deg(x_i)$, but $\deg(x_i)$ may go up to 3 by §3.3.3, so we apply techniques from §2.4.2 to compute with it.

Observe that $h = R_{f2}R_{f3}R_{f4} \in \mathbb{Q}[x]$ is a square-free polynomial whose zeroes in $\mathbb{C}$ are precisely the x-coordinates of the s singular points of f in $\mathbb{C}^2$. (The x-coordinate of each singularity has multiplicity at least 2 as a zero of R_f, but at most 4, since $\deg(R_f) \leq 6$.) In particular, $\deg(h) = s$.

Let us first assume that h is irreducible. Let $\vartheta \in \mathbb{C}$ be any of its s complex zeroes. (One of them is x_i, but we prefer the symbol ϑ to express the freedom of choosing any conjugate.) Then $\mathbb{Q}(\vartheta) \simeq \mathbb{Q}[x]/(h)$, see §2.4.1. The polynomial $\varphi(y) = f(\vartheta, y) \in \mathbb{Q}(\vartheta)[y]$ corresponds to $\overline{f} \in (\mathbb{Q}[x]/(h))[y]$. Since f is y-regular, the leading coefficient is a non-zero element of $\mathbb{Q}$ and hence unaffected by computing modulo h, so that $\deg(\varphi) = \deg(f) = 3$. In the same way, f_y represents $\varphi'(y)$ and $\deg(\varphi') = \deg(f_y) = 2$.



We want to factor $\varphi(y) = \ell(f)(y+\eta_1)(y+\eta_2)^2$ by multiplicities. Let us start by computing $\gcd(\varphi, \varphi')$. The first step in the Euclidean Algorithm is to compute the remainder of φ modulo φ'. In terms of representatives f and f_y this means to consider them as elements of $Q(\mathbb{Q}[x])[y]$ and perform univariate polynomial division of f by f_y. Since $\ell(f_y) \in \mathbb{Q}$ is a unit in $\mathbb{Q}[x]$, we can actually perform this division within $\mathbb{Q}[x][y]$. Let the resulting remainder be $g(x,y) = \alpha(x)y + \beta(x) \in \mathbb{Q}[x][y]$.

If $\eta_1 \neq \eta_2$, as required by nonverticality, $\deg(\gcd(\varphi, \varphi')) = 1$, so that the polynomial $\overline{g} \neq \overline{0}$ in $(\mathbb{Q}[x]/(h))[y]$ corresponds (up to constants) to the monic linear polynomial $\gcd(\varphi, \varphi')$ in $\mathbb{Q}(\vartheta)(y)$. Conversely, $\eta_1 = \eta_2$ implies $\deg(\gcd(\varphi, \varphi')) = 2$ so that $\overline{g} = \overline{0}$.

Hence we try to invert $\alpha \in \mathbb{Q}[x]$ modulo h by an extended gcd computation (§2.2.2, §2.4.2) and complain about a vertical singularity if it fails. If it works, the monic polynomial $\varphi_2(x,y) := \alpha^{-1}g = y + \eta_2(x) \in \mathbb{Q}[x][y]$ represents $\gcd(\varphi, \varphi') = y + \eta_2$. Dividing f by $\ell(f)\varphi_2^2$ yields the representative $\varphi_1(x,y) = y + \eta_1(x) \in \mathbb{Q}[x][y]$ of the other square-free factor $y + \eta_1$ of φ.

The polynomials $-\eta_1(x), -\eta_2(x) \in \mathbb{Q}[x]$ represent the two zeroes of $\varphi \in \mathbb{Q}(\vartheta)[y]$. Let $\delta(x) = (-\eta_2(x)) - (-\eta_1(x))$ be their difference. The crucial point of the whole computation is that substituting x_i, which is one of the zeroes ϑ of h, for x in $\delta(x)$ yields $\delta(x_i) = y_i - y_i'$. In this way, evaluating δ allows us to compute $\operatorname{sign}(y_i - y_i')$, as desired.

It remains to argue that the same technique applies to the case that h is not irreducible. This issue has been discussed in §2.4.2. The only operation critical in this respect is the modular inversion of α. If $d := \gcd(\alpha, h)$ is nonconstant, some singularity $(\vartheta,\eta) \in \mathbb{C}^2$ of f is vertical, viz. one with $d(\vartheta) = 0$. If (ϑ,η) is real, this constitutes a violation of the nonverticality condition and justifies the abortion of the algorithm. The case that (ϑ,η) is not real cannot occur, since we would be in the situation of a triangle with two complex vertices, where all tangents of f at (ϑ,η) are components of f and automatically nonvertical by the y-regularity condition.

Let us summarize:

```
**Delta:** Let $h := R_{f2}R_{f3}R_{f4} \in \mathbb{Q}[x]$, $\deg(h) > 1$.
Do Euclidean Division $f = qf_y + g$ in $\mathbb{Q}[x][y]$ to define $q$ and $g = \alpha y + \beta$.
Compute $d,u,v \in \mathbb{Q}[x]$ such that $d = \gcd(\alpha,h) = u\alpha + vh$.
If $d \neq 1$, signal "vertical singularity" and abort.
Let $\varphi_2 = ug \bmod h = y + \eta_2(x)$ and $\varphi_1 = f/(\ell(f)\varphi_2^2) \bmod h = y + \eta_1(x)$.
Let $\delta = \eta_1(x) - \eta_2(x) \in \mathbb{Q}[x]$.
```

We return to the problem of analyzing a singularity $(x_i,y_i) \in \mathbb{R}^2$ of f. If $s = 2$, then both singularities are real and the zero x_i of h is known as an expression involving at most one square root. So the techniques of §2.4.3 allow us to evaluate $\operatorname{sign}(\delta(x_i))$ straight away. Since there is a real line component joining the two real singularities, it is a crunode.

If $s = 3$, then f is a triangle. Let us first consider the case that all vertices are real and have x-coordinates $x_i < x_j < x_k$. It is obvious from the shape of a triangle that $\operatorname{sign}(\delta(x_i)) \neq \operatorname{sign}(\delta(x_j)) \neq \operatorname{sign}(\delta(x_k))$. But we have $\deg(\delta) \leq \deg(h) - 1 = 2$, so that $\operatorname{sign}(\delta(x_i)) = \sigma$, $\operatorname{sign}(\delta(x_j)) = -\sigma$, and $\operatorname{sign}(\delta(x_k)) = \sigma$ for $\sigma := \operatorname{sign}(\ell(\delta))$. Hence σ contains the complete answer if all vertices are real!

In fact, the same holds if just one vertex is real, but this requires a bit of calculation. First observe $\delta \in \mathbb{C}[x]$ is determined uniquely as the univariate polynomial of degree ≤ 2 whose graph interpolates through the complex points $(x_i,\, y_i - y'_i)$, $(x_j,\, y_j - y'_j)$ and $(x_k,\, y_k - y'_k)$. Hence δ transforms in the obvious way under translation and under scaling of the coordinate system, and it is a well-defined object for all y-regular triangles $f \in \mathbb{C}[x,y]$.

Assume that $f \in \mathbb{R}[x,y]$ is a triangle with two complex vertices. Translate the coordinate system so that the one real vertex is $(0,0)$. Factor $f = gh$ into a line $g \in \mathbb{R}[x,y]$ and a complex line pair $h = \alpha y^2 + \beta xy + \gamma x^2 \in \mathbb{R}[x,y]$. Since a complex line pair is irreducible in $\mathbb{R}[x,y]$, we have $\beta^2 - 4\alpha\gamma < 0$, so that $\alpha\gamma > 0$. The freedom to replace h by a constant multiple and to scale the x-axis by a positive factor allows us to transform h to $h = y^2 + bxy + x^2$ with discriminant $b^2 - 4 < 0$. The nonvertical real line g can be normalized to have $\ell(g) = 1$, meaning $g(x,y) = y + ax + c$ where $c = g(0,0) \neq 0$. The reader is invited to follow the algorithm **Delta** for f and verify with the help of a computer algebra system that

$$\delta(x) = \frac{3}{c}(a^2 - ba + 1)x^2 + (4a - 2b)x + c. \tag{4.2}$$

(The author regrets that he has not found a more elegant argument.)

Consider the factor $\lambda(a) = a^2 - ba + 1$ of $\ell(\delta)$ as a univariate polynomial in a. Its discriminant $b^2 - 4$ equals the discriminant of h and is hence negative, so that $\lambda(a)$ has no real zeroes and is therefore positive for all a. Thus $\operatorname{sign}(\ell(\delta)) = \operatorname{sign}(c)$, where $c = \delta(0)$ is precisely the difference of y-coordinates between the real singularity $(0,0)$ of f and the point $(0,-c)$ on the continuing arc.

To derive this result, we have translated the coordinate system by a real vector, and we have scaled the x-axis by a positive real factor. Both kinds of transformation leave $\operatorname{sign}(\ell(\delta))$ invariant. This proves for any y-regular triangle $f \in \mathbb{R}[x,y]$ with the unique real vertex (x_i,y_i) that $\operatorname{sign}(\delta(x_i)) = \operatorname{sign}(\ell(\delta))$.

We are now ready to state the resulting part of the algorithm:

Multiple Singularities: Let $(x_i,y_i) \in \mathbb{R}^2$ be a singularity.
Let f have altogether $s > 1$ singularities in $\mathbb{C}^2$, $r \geq 1$ of them real.

Compute the polynomial $\delta \in \mathbb{Q}[x]$.

If $s = 2$, then (x_i,y_i) is a crunode, involving two arcs on either side.
The continuing arc runs above the singularity iff $\delta(x_i) < 0$.

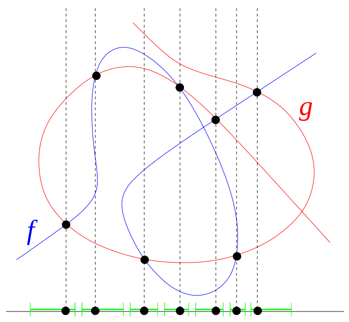
If $s = 3$, (x_i,y_i) is the i-th singularity in ascending x-order.
If $r = 3$, (x_i,y_i) is a crunode, else $r = 1$ and it is an acnode.
The continuing arc runs above the singularity iff $(-1)^i\ell(\delta) > 0$.

For $s = 3$, the distinction between crunode and acnode can also be made by checking whether $k_i = k_{i+1} = 3$ or $k_i = k_{i+1} = 1$.

4.2 Analyzing Two Curves

4.2.1 Basics

Let us turn to the geometric analysis of a pair $\{f,g\} \subseteq \mathbb{Q}[x,y]$ of real algebraic curves with degrees ≤ 3. We take the same point of view as in the analysis of one curve and ask: For a given x-coordinate $\xi \in \mathbb{R}$, what is the number and relative position of points of f and g over ξ, and how does this change as we vary ξ?



The algorithmic answer to this question is similar in style to the analysis of one curve, which was already a special kind of intersection problem. In particular, it follows the general ideas outlined in §4.1.1.

We can restrict ourselves to pairs of coprime curves, since this is a sufficient basis for handling sweepable segments of curves (see Section 5.2), and that in turn is sufficient to analyze arbitrary finite sets of curves or general segments (see Section 5.3). So let f and g henceforth be coprime. As always, we also demand them to be y-regular.

To apply the results of §3.2.5 to the point set $f \cup g$, note that $f \cup g = fg$ by (3.10). It follows that $f \cup g$ consists of a finite number of arcs of fg, each of which is a subset of an arc of either f or g. Referring to this, we call an arc of fg either an f-arc or a g-arc. The arcs of fg are linked at the finitely many *(two-curve) event points* $v \in fg \cap (fg)_y$ of $\{f,g\}$. The following proposition elucidates the origin of two-curve events.

Proposition 4.2.1: *Let K be a field and let $f,g \in K[x,y]$ be algebraic curves. Then*

$$fg \cap (fg)_y = (f \cap f_y) \cup (g \cap g_y) \cup (f \cap g) \tag{4.3}$$

Proof: Observe $(fg)_y = f_y g + f g_y$. From this, "$\supseteq$" follows immediately. For "$\subseteq$" let $v \in fg \cap (fg)_y$ and assume w. l. o. g. that $f(v) = 0$. Then $0 = (fg)_y(v) = (f_y g)(v)$ so that $v \in f \cap f_y$ or $v \in f \cap g$. $\square$

Any intersection point $v \in f \cap g$ is an event point of $\{f,g\}$. It may or may not be a one-curve event of f or g.

If an event point v of $\{f,g\}$ is not an intersection point, it is a one-curve event point of either f (that is $v \in f \cap f_y$) or g (that is $v \in g \cap g_y$).

The rest of this section describes an algorithm that takes a pair $\{f,g\}$ of algebraic curves $f,g \in \mathbb{Q}[x,y]$ of degrees ≤ 3, subject to certain conditions, and determines:

- The decomposition of the x-axis into two-curve event x-coordinates and open intervals between them.
- The number and relative position of arcs of f and g over any $\xi \in \mathbb{R}$.
- For each two-curve event: Whether it is an intersection; and what kind of one-curve event it is on f and g, if any.
- The intersection multiplicity of each intersection.
- The arcs involved in each event, and the sorted sequence of arcs below and above the event.

The conditions imposed on f and g, besides the degree bound, are as follows:

- f and g both satisfy the conditions of §4.1.1.
- f and g are coprime.
- No two points of $V_{\mathbb{C}}(f) \cap V_{\mathbb{C}}(g)$ are covertical.
- No two event points of $\{f,g\}$ are covertical.
- No point of $f \cap g$ is an x-extreme point of f or g.

- The Jacobi curve $J = f_x g_y - f_y g_x$ of $\{f,g\}$ is y-regular, and its set of one-curve events $V_{\mathbb{C}}(J) \cap V_{\mathbb{C}}((J/\gcd(J,J_y))_y)$ does not contain a point covertical to a $v \in \mathbb{R}^2$ with $\mathrm{mult}(v; f,g) = 2$, see §4.2.5.

The role of these conditions is identical to the conditions for the analysis of one curve (§4.1.1): The algorithm checks whether they hold to the extent it relies on them and signals violations. Establishing coprimality has been discussed above. For the other conditions, see Section 4.3.

4.2.2 Event x-coordinates

Analogous to §4.1.2, the sorted sequence of two-curve event x-coordinates induces a partition of the x-axis, and again we use terms like "interval between events" etc.

Each two-curve event is an intersection and/or a one-curve event. The latter are known from the analysis of one curve. The former correspond to real zeroes of $\mathrm{res}(f,g,y)$ by Proposition 2.3.3. Once the noncoverticality of intersections is guaranteed (we check it in the analysis of each intersection), we know this correspondence of real zeroes and real intersections to be bijective by Proposition 2.3.5.

Recall that by Propositions 2.2.1 and 2.3.4, f and g are coprime iff $\mathrm{res}(f,g,y) \neq 0$.

Two-curve Event x-Coordinates:
Invoke the analysis of one curve for f and g.

Compute $R_{fg} := \mathrm{res}(f,g,y)$. If $R_{fg} = 0$, signal "not coprime" and abort.
Factor R_{fg} by multiplicities and obtain $R_{fg} = \prod_{m=1}^{M} R_{fgm}^m$.

Merge the real zeroes of all square-free factors of R_{fg}, R_f and R_g to yield the sorted sequence of two-curve event x-coordinates $x_1 < x_2 < \ldots < x_n$ and record their respective multiplicities $m_i^{(fg)}$, $m_i^{(f)}$ and $m_i^{(g)}$.

If there is $1 \leq i \leq n$ such that $m_i^{(fg)} = 0 \ \wedge \ m_i^{(f)} > 0 \ \wedge \ m_i^{(g)} > 0$,
signal "covertical events" and abort.

For intervals I between events, the analysis of $B = fg \cap I \times \mathbb{R}$ consists of sorting its connected components $B_1, \ldots, B_{k_i}$ in ascending y-order and distinguishing between f-arcs and g-arcs. As in §4.1.2 we say that the arc $A \supseteq B_\mu$ of f or g has *arc number μ on fg over I,* or is the *μ-th arc of f over I.*

Arcs over Intervals: Using the rational isolating intervals of the x_i, determine a rational point $r_i \in]x_{i-1}, x_i[\cap \mathbb{Q}$ for each interval between events.

Find and sort the real zeroes of $f(r_i,y), g(r_i,y) \in \mathbb{Q}[y]$ (as in §2.4.4) to determine the sorted sequence of f-arcs and g-arcs over I.

The rest of this section is concerned with the analysis of event points.

4.2.3 One-curve events

Let x_i be the x-coordinate of a two-curve event such that $m_i^{(fg)} = 0$. Then x_i originates from a one-curve event (x_i,y_i). Let us assume w. l. o. g. that $(x_i,y_i) \in f$. The absence of an intersection over x_i implies $(x_i,y_i) \notin g$. **Two-curve Event x-Coordinates** has checked that there is no one-curve event of g over x_i. Hence the analysis of $f \cup g$ over x_i consists of locating (x_i,y_i) between the continuing arcs of f and g.

If the event (x_i,y_i) is not an isolated point of f, then its position relative to the arcs of f and g can be inferred from the position of the arcs of f containing (x_i,y_i).

Now assume the event (x_i,y_i) is an isolated point of f. Such an acnode can occur

- on a singular irreducible cubic,
- on a triangle,
- on a line pair.

In each case, f_y is square free and does not have a one-curve event at x_i. In case of a line pair, this follows from $\deg(f_y) = 1$. In the other cases, this follows from $\deg(f_y) = 2$ in conjunction with the fact that there must be a non-event point (x_i,y_i') on f (see §4.1.6) and, by the Mean Value Theorem, an arc of f_y between (x_i,y_i') and (x_i,y_i). So we can use f_y as a curve containing (x_i,y_i) in the interior of one of its two arcs to locate this point. The relative position of (x_i,y_i) and (x_i,y_i'), known from the analysis of f, identifies the arc of f_y containing (x_i,y_i).

If f_y shares a component with g, this component does not contain (x_i,y_i) and can be ignored.

One-curve Events: Let $m_i^{(fg)} = 0$, (x_i,y_i) an event on f.

If (x_i,y_i) is an event with an involved arc extending over an interval I incident to x_i (i. e. anything but an acnode), iterate through the sequence of arcs over I. The arcs seen before the lowest involved arc and beyond the highest involved arc form the sequences of arcs below and above the event.

The rest applies to the case that (x_i,y_i) is an acnode.

Refine the isolating interval $]r_i,r_{i+1}[$ of x_i to an interval $]r_-,r_+[\ni x_i$ so that $[r_-,r_+]$ contains no zero of $\operatorname{res}(f_y, f_{yy}, y)$.

Let $h := f_y$, $R := \operatorname{res}(g,h,y)$.
If $R = 0$, let $h := f_y / \gcd(g, f_y)$, $R := \operatorname{res}(g,h,y)$.
Refine $[r_-,r_+]$ further until it contains no zero of R different from x_i.

Let μ be the number of the arc of h containing (x_i,y_i):
If $\deg(h) = 1$, then $\mu = 1$. Otherwise $\deg(h) = 2$ and $\mu = \begin{cases} 1 & \text{if } y_i < y_i' \\ 2 & \text{if } y_i > y_i' \end{cases}$

Locate the μ-th real zero of $h(r_-,y)$ among the real zeroes of $g(r_-,y)$ to locate (x_i,y_i) between the arcs of fg.

For a non-acnode, it is preferable to locate it using the supporting arcs because this allows us to infer the desired information from the already available analysis of the neighbouring intervals without much further computation.

The complicated procedure for an acnode is motivated by the intention to avoid explicit arithmetic in x_i since $\deg(x_i) = 3$ for an irrational acnode in a triangle. A possible optimization is to distinguish the case $x_i \in \mathbb{Q}$ and handle it by inspecting the curves at x_i with explicit arithmetic in x_i.

4.2.4 Intersections in general

Let x_i be the x-coordinate of a two-curve event such that $m_i^{(fg)} > 0$. By the conditions on $\{f, g\}$, there must not be an x-extreme point of f or g over x_i: It may neither be covertical nor equal to the intersection point.

> **Exclude x-Extreme:**
> If there is $1 \leq i \leq n$ such that $m_i^{(fg)} > 0$ and $m_i^{(f)} = 1$ or $m_i^{(g)} = 1$,
> signal "x-extreme over intersection x-coordinate" and abort.

So let us assume from now on that this has been checked and that one-curve events of f and g over x_i, if any, are singularities.

If both f and g have a singularity over x_i, the situation is special insofar that we have explicit y-coordinates for all points from the analyses of one curve and that the noncoverticality condition requires the intersection point to be equal to both singularities. We handle this case in §4.2.8.

In the remaining cases, at least one of the polynomials $f|_{x_i}, g|_{x_i} \in \mathbb{R}[y]$ is square free by Corollary 3.2.10 so that $d := \deg(\gcd(f|_{x_i}, g|_{x_i}))$ is the number of distinct common zeroes $y \in \mathbb{C}$ of $f(x_i, y)$ and $g(x_i, y)$. Hence the noncoverticality condition for intersection points holds iff $d = 1$. By Corollary 2.3.7, this is equivalent to $\operatorname{sres}_1(f, g, y)(x_i) \neq 0$. Thus we can check this condition without explicit arithmetic in x_i by inspecting the gcd of the subresultant and the resultant factor defining x_i. Once noncoverticality holds, the intersection multiplicity is known to be $m_i^{(fg)}$.

The analysis of intersections splits into the following cases:

- Neither curve has a singularity over x_i – §4.2.5.
- Exactly one of the curves has a singularity over x_i, and x_i is known to be rational, in particular by uniqueness of the singularity – §4.2.6.
- Exactly one of the curves has a singularity over x_i and x_i is not known to be rational – §4.2.7.
- Both curves have a singularity over x_i – §4.2.8.

For simplicity, the descriptions of the two asymmetric cases assume that the singularity is on f.

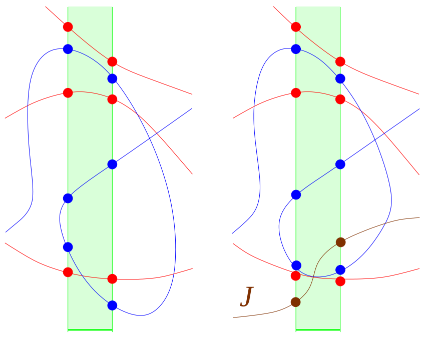
The main task in analyzing an intersection is to determine which arcs of f and g, as identified by their arc numbers, are involved. Except for intersections in isolated points, this together with the analysis over neighbouring intervals completely determines the geometry of $f \cup g$ over x_i.

4.2.5 Intersection regular-regular

Let x_i be a two-curve event x-coordinate with $m := m_i^{(fg)} > 0$ and $m_i^{(f)} = m_i^{(g)} = 0$. After checking the noncoverticality condition using the subresultant (cf. §4.2.4), we know there is exactly one intersection point of f and g and no one-curve event of either curve over x_i.

By Corollary 3.2.12, the intersection over x_i causes the intersecting arcs to change sides iff m is odd. In that case, the two intersecting arcs are directly discernible from the arc sequences over the incident intervals.

If m is even, the arc sequences over the incident intervals are equal. However, since $\deg(R_{fg}) \leq 9$ by Corollary 3.2.5, the square-free factor R_{fgm} defining x_i has degree at most 2 for $m = 4$ and degree 1 for $m = 6$ and $m = 8$, allowing us to compute explicitly with its zero x_i in $\mathbb{Q}$ or $\mathbb{Q}(\sqrt{D})$, $D > 0$.



Intersections with odd (left) and even (right) multiplicity m. The Jacobi curve J is introduced below.

We first state the algorithm for these easy cases.

Intersection regular-regular: Let $m := m_i^{(fg)} > 0$ and $m_i^{(f)} = m_i^{(g)} = 0$.

If x_i is a zero of $\mathrm{sres}_1(f, g, y)$, signal "covertical events" and abort.

If m is odd, iterate simultaneously through the arc sequences over $]x_{i-1}, x_i[$ and $]x_i, x_{i+1}[$ until a transposition of two adjacent arcs occurs. These two arcs intersect.

If $m = 4, 6, 8$, obtain an explicit representation of x_i from R_{fgm}.
Compute y_i as the zero of the linear polynomial $d := \gcd(f|_{x_i}, g|_{x_i})$.
Compute the real zeroes of $f|_{x_i}/d$ and $g|_{x_i}/d$.
Sort them and locate y_i between them to determine the sequence of arcs.

(For $m = 2$ see below.)

If $m = 1$, there are no covertical intersections over x_i and the subresultant check can be skipped.

If $m = 4$ and x_i is irrational so that root expression arithmetic is required, first use a symbolic representation to compute d and $f|_{x_i}/d$, $g|_{x_i}/d$; then convert to a separation bound number type to facilitate the nested root extraction needed for solving $f|_{x_i}/d = 0$, $g|_{x_i}/d = 0$ and to enable an efficient comparison of zeroes (see §2.4.3).

The remaining case $m = 2$ can be tackled using the Jacobi curve, a technique introduced by Nicola Wolpert and Elmar Schömer [GH$^+$01] [W02] [W03]. The *Jacobi curve* J of f and g is the determinant of the Jacobi matrix of the map $(f,g)\colon \mathbb{R}^2 \to \mathbb{R}^2$, that is

$$J := \det(\nabla f,\, \nabla g) = f_x g_y - f_y g_x. \tag{4.4}$$

The zero set of the polynomial J consists of those points $v \in \mathbb{R}^2$ for which $\nabla f(v)$ and $\nabla g(v)$ are collinear. For $v \in f \cap g$ this is equivalent to $\mathrm{mult}(v; f,g) \geq 2$ by Proposition 3.2.6.

It remains to argue that J deserves the name "curve", meaning that J is non-constant. This does not always hold, e. g. when $f = y^2 - 1$ and $g = y + 2$. It does hold, however, if there is a point $v \in \mathbb{R}^2$ not lying on a common component of f and g such that $\mathrm{mult}(v; f) = \mathrm{mult}(v; g) = 1$ and $\mathrm{mult}(v; f,g) \geq 2$.

We only sketch a proof: Clearly $J(v) = 0$ so J is not a non-zero constant. Assume $J = 0$. Then ∇f and ∇g are nonzero and collinear around v. Consider the vector fields $\nabla f^\perp$ and $\nabla g^\perp$ obtained by taking perpendicular vectors. Parametrizations γ_f and γ_g of f and g around v with $\gamma_f(0) = \gamma_g(0) = v$, reparametrized to suitable speeds, are the unique integral curves of these vector fields, that is $\gamma_f{}'(t) = \nabla f(\gamma_f(t))^\perp$ and $\gamma_g{}'(t) = \nabla g(\gamma_g(t))^\perp$ (see [K95, 12.3] [K97, 4.2.II]). Due to the collinearity of the vector fields, γ_f and γ_g differ by speed *only*, so that their images coincide around v in infinitely many points, contradicting the premise that v does not lie on a common component.

The claim also follows from the proof of Proposition 4.2.3 referred to below.

Consequently, we may speak of the Jacobi *curve* of f and g in the present situation where (x_i, y_i) can play the role of v. Next we demonstrate that the Jacobi curve does not depend on the choice of a specific coordinate system.

Proposition 4.2.2: *Let $f, g \in \mathbb{C}[x,y]$ be two algebraic curves, and let J be their Jacobi curve. Let A be an affine change of coordinates. Then the Jacobi curve of $f \circ A$ and $g \circ A$ is $J \circ A$.*

Proof: We consider translations and linear changes of coordinates separately. The claim then follows by composition.

For a translation $A(v) = v + b$, we have $(f \circ A)_x(v) = f_x(v + b) = (f_x \circ A)(v)$ etc. by the chain rule, proving the claim.

For a linear change of coordinates $A(v) = Mv$, applying the chain rule shows that $\nabla(f \circ M)(v) = M^\mathsf{T}((\nabla f)(M(v)))$. Consequently,

$$\begin{aligned}\det\Big(\Big(\nabla(f \circ M),\ \nabla(g \circ M)\Big)\Big) &= \det\Big(M^\mathsf{T}\Big((\nabla f) \circ M,\ (\nabla g) \circ M\Big)\Big) \\ &= \det(M) \cdot J \circ M\end{aligned}$$

where $\det(M)$ is a non-zero constant. □

The practical use of Jacobi curves comes from the following result.

Proposition 4.2.3: *Let $f,g \in \mathbb{Q}[x,y]$ be two algebraic curves having an intersection point $v \in \mathbb{C}^2$ with multiplicity* $\mathrm{mult}(v; f,g) = 2$. *Let J be their Jacobi curve. Then* $\mathrm{mult}(v; f,J) = \mathrm{mult}(v; g,J) = 1$.

Proof: A proof by resultant matrix computations under certain position assumptions is given in [W02, Thm. 4.9]. It transfers to general f,g after a suitable change of coordinates. □

The proposition allows us to locate the arcs involved in an intersection (x_i,y_i) of multiplicity $m = 2$ by detecting the side change of an arc of J with the two intersecting arcs of f and g, provided that we can find an interval $I \ni x_i$ such that the arc of J containing (x_i,y_i) extends over both boundaries, and such that there are no events of J, of $\{J,f\}$ and of $\{J,g\}$ in $I \times \mathbb{R}$ except for the intersections in (x_i,y_i). In principle, this means that the event-describing resultants have no zeroes except for the simple zero coming from the intersection in (x_i,y_i). However, the resultants may vanish due to common factors. Also, it is not clear that J is y-regular, and we demand this explicitly.

To begin with, consider $R_J := \mathrm{res}(J,J_y,y)$. If $R_J = 0$, then J is not square free, we replace J by $J/\gcd(J,J_y)$, and compute $R_J \neq 0$ again. We demand $R_J(x_0) \neq 0$. This condition means that there is no one-curve event of J over x_i, in particular (x_i,y_i) is not an x-extreme point.

Now consider $R_{Jf} := \mathrm{res}(J,f,y)$ and $R_{Jg} := \mathrm{res}(J,g,y)$. Their treatment is symmetric, and we describe it for R_{Jf}. If $R_{Jf} = 0$, then J and f have a non-constant common factor $h = \gcd(J,f)$. Since $\mathrm{mult}((x_i,y_i);J,f) = 1$, we have $(x_i,y_i) \notin h$ and we can replace J by J/h. We demand the multiplicity of x_i as a zero of R_{Jf} to be 1, which is equivalent to the absence of intersections covertical to (x_i,y_i).

Finally we have to determine an interval I around x_i not containing any zero of R_J and not containing any zero of R_{Jf} and R_{Jf} other than x_i. At its boundaries, we can observe the desired jump of J over the intersecting arcs. Besides this jump, the arc sequences are equal.

To get a rough idea of the effort involved in handling J, observe the obvious degree bound $\deg(J) \leq \deg(f) + \deg(g) - 2 \leq 4$.

Intersection regular-regular, subcase m=2:

Compute $J := f_x g_y - f_y g_x$. If J is not y-regular, abort.

Let $R_J := \mathrm{res}(J, J_y, y)$.
If $R_J = 0$, let $J := J/\gcd(J, J_y)$ and recompute R_J.

Let $R_{Jf} := \mathrm{res}(J, f, y)$.
If $R_{Jf} = 0$, let $J := J/\gcd(J, f)$ and recompute R_J, R_{Jf}.

Let $R_{Jg} := \mathrm{res}(J, g, y)$.
If $R_{Jg} = 0$, let $J := J/\gcd(J, g)$ and recompute R_J, R_{Jf}, R_{Jg}.

If x_i is a zero of R_J or a multiple zero of R_{Jf} or R_{Jg},
signal "covertical Jacobi event" and abort.

Refine the isolating interval $]r_i, r_{i+1}[$ of x_i to an interval $[r_-, r_+] \ni x_i$ containing no zero of $R_J R_{Jf} R_{Jg}$ except x_i.

Compute the sorted sequences of real zeroes of $f(r_-, y), g(r_-, y), J(r_-, y)$ and $f(r_+, y), g(r_+, y), J(r_+, y)$ and compare them to detect the pair of an f-arc and a g-arc that changes sides with a J-arc.

It is arguable whether one should recompute non-zero resultants or not. Recomputing takes time but makes computing $[r_-, r_+]$ cheaper and may avoid an unneeded abortion of the algorithm.

4.2.6 Intersection regular-singular, rational case

Let x_i be a two-curve event x-coordinate with $m_i^{(fg)} > 0$, $m_i^{(f)} \geq 2$, and $m_i^{(g)} = 0$. This means there is an intersection point of f and g, a singularity of f and no one-curve event of g over x_i.

Furthermore, let x_i be known as a rational number, which is certainly the case if f has a unique singularity in $\mathbb{C}^2$. It can also happen via simplifications of the defining polynomial of x_i (see §2.4.4) in comparisons with numbers from other sources, typically because the analyses of different curve pairs are performed in an interleaved fashion during arrangement computation.

The noncoverticality condition requires that singularity and intersection point are equal, and that there is only one intersection. We have to check this. Furthermore, we have to determine the arc of g involved in the intersection.

Factorization of $f(x_i, y) \in \mathbb{Q}[y]$ by multiplicities yields the singularity's y-coordinate $y_i \in \mathbb{Q}$ and, if present, the y-coordinate y_i' of a continuing arc of f. The noncoverticality condition holds iff $g(x_i, y_i') \neq 0$ or y_i' does not exist.

The continuing arcs of g correspond to zeroes of $g(x_i, y)/(y - y_i) \in \mathbb{Q}[y]$ which has degree at most 2. The sorted sequence consisting of these zeroes and y_i, y_i' completely describes the geometry of $\{f, g\}$ over x_i.

Intersection regular-singular rational:
Let $x_i \in \mathbb{Q}$, $m_i^{(fg)} > 0$, $m_i^{(f)} \geq 2$, and $m_i^{(g)} = 0$.

Let (x_i, y_i) be the singularity of f over x_i.
Let (x_i, y_i') be the other point of f over x_i, if any.

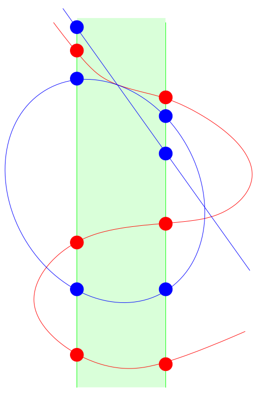
If y_i' exists and $g(x_i, y_i') = 0$, signal "covertical events" and abort.

Compute the sorted sequence consisting of y_i, y_i' (if exists) and the real zeroes of $g(x_i, y)/(y - y_i)$ (if any) to determine the sequence of arcs below and above the intersection (x_i, y_i).

4.2.7 Intersection regular-singular, algebraic case

Let x_i be an event x-coordinate with $m_i^{(fg)} > 0$, $m_i^{(f)} \geq 2$, and $m_i^{(g)} = 0$. This means there is an intersection point of f and g, a singularity of f and no event of g over x_i. Let x_i be represented with defining polynomial of degree ≥ 2. Then the singularity of f is not unique in $\mathbb{C}^2$ and hence is an acnode or a crunode.

After checking the subresultant (cf. §4.2.4), we know there is exactly one intersection point of f and g over x_i. It remains to check that the intersection occurs in the singularity and to determine which arc of g is involved.



In a crunode v of f, two branches of f intersect. We have seen in §3.3.4 that both branches have an analytic parametrization of the form $x \mapsto (x, \varphi(x))$ such that the slopes $\varphi'(x_i)$ differ (because f has two distinct tangents at v). An arc of g passing through the crunode also has a parametrization of this form by §3.2.5, and its slope can coincide with at most one of the two branches of f. Thus it follows from the arguments used in Corollary 3.2.12 that such a g-arc changes sides with at least one of the f-arcs, and we know the f-arcs to change sides with each other.

It follows further that a g-arc A intersecting f in the crunode v fulfills at least one of the following conditions:

(i) On the left-hand side, A lies between the two arcs of f containing v.

(ii) On the right-hand side, A lies between the two arcs of f containing v.

(iii) On one side, A lies below and on the other side, A lies above the respective two arcs of f containing v.

It is easy to see that the converse also holds: If one of these conditions holds for A, it intersects f in v.

Hence comparing arc sequences allows us to check that the intersection happens in the singularity, and if so, to determine the g-arc involved. Together with the analysis over the incident intervals, this describes the geometry of $f \cup g$ over x_i completely.

> **Intersection regular-singular algebraic crunode:**
> Let (x_i, y_i) be a crunode on f and $m_i^{(fg)} > 0$, $m_i^{(g)} = 0$.
>
> If x_i is a zero of $\mathrm{sres}_1(f, g, y)$, signal "covertical events" and abort.
>
> Inspect the arc sequences over incident intervals.
> If a g-arc fulfills one of (i)–(iii), it is the g-arc intersecting f in (x_i, y_i).
> If no such g-arc exists, signal "covertical events" and abort.

If $m_i^{(fg)} = 1$, then the algorithm can signal covertical events immediately, since an intersection in a singularity has multiplicity at least 2 by Proposition 3.2.6.

Handling an acnode of f is considerably harder, since it does not have a supporting arc. An acnode which is not known to be rational can only occur for a triangle f. Using f_y as an auxiliary curve supporting the acnode similar to §4.2.3 seems possible in principle. In practice, however, the author has given up on handling the case distinction of how f_y may intersect g in the acnode.

So we resort to explicit arithmetic, which is especially complicated in this case since we have $\deg(x_i) = 3$. Except for the number types, the algorithm is similar to §4.2.6.

> **Intersection regular-singular algebraic acnode:**
> Let (x_i, y_i) be an acnode on f and $m_i^{(fg)} > 0$, $m_i^{(g)} = 0$.
> Let (x_i, y_i') be the non-event point on f covertical to (x_i, y_i).
>
> If $g(x_i, y_i') = 0$, signal "covertical events" and abort.
>
> Compute the sorted sequence consisting of y_i, y_i' and the real zeroes of $g(x_i, y)/(y - y_i)$ (if any) to determine the sequence of arcs below and above the intersection (x_i, y_i).

There are basically two choices of how to perform arithmetic: Either use exclusively the symbolic method of §2.4.2, or combine it with a separation bound number type of §2.4.3 strong enough to express x_i, that is `leda::real` with the diamond operator.

In either case, we start symbolically, using $h := R_{fm_i^{(f)}}$ as the defining polynomial for x_i. In §4.1.6 we have already computed y_i' and y_i as polynomials $-\eta_1(x_i)$ and $-\eta_2(x_i)$. Performing the univariate polynomial division $\chi(y) := g(x_i, y)/(y - y_i)$ amounts to dividing $g/(y + \eta_2)$ in $\mathbb{Q}[x][y]$ and reducing the coefficients of y^i modulo h. By y-regularity, we have $\deg(\chi) = \deg(g) - 1$.

The test $g(x_i, -\eta_1(x_i)) = 0$ is straightforward to perform with either symbolic or separation bound arithmetic. (We expect that equality does not hold.)

Finding the zeroes of χ and comparing them to y_i, y'_i is easy with separation bound arithmetic, since $\deg(\chi) \leq 2$: If χ is constant, there is nothing to do. If it is linear, just solve for the unique zero by division. If it is quadratic, evaluate the discriminant (which is non-zero) and write down the two zeroes, if any, using the square root operator. Then compare to sort.

With symbolic arithmetic, there are two complications: For $\deg(\chi) = 2$ we cannot in general express the real zeroes of $\chi(y) = c_2y^2 + c_1x + c_0$ (if there are any at all) as elements of $\mathbb{Q}[x]/(h)$ to compare them. Instead, we compare them implicitly to, say, y_i in the following fashion: If $\operatorname{sign}(\chi(y_i)) \neq \operatorname{sign}(c_2)$, then y_i lies between the two zeroes of χ. Otherwise, $\operatorname{sign}(\frac{c_1}{2c_2} - y_i)$ indicates whether y_i is less than or greater than both zeroes of χ. So we only need three-valued comparisons on $\mathbb{Q}[x]/(h)$ (fixing x_i as the desired zero of h).

In §2.4.2 we have described how to implement a number type for arithmetic in $\mathbb{Q}[x]/(h)$ supporting these operations, including the handling of a non-irreducible modulus h (be it by prior factorization into irreducibles exploiting $\deg(h) \leq 3$, be it by coping with a factorization obtained during calculations) and three-valued comparisons. However, we do not need it in other places, and a thorough implementation seemed duplicate effort to the author, since the diamond operator is coming into existence, and all the numerics necessary to implement three-valued comparisons on $\mathbb{Q}[x]/(h)$ are a part of its implementation. On the other hand, the diamond operator has not been available in time to be properly integrated into the implementation.

Therefore, the author has decided leave out the very special case of an intersection in a triangle's irrational acnode in the version of the implementation that has been prepared for this thesis. The case is detected correctly but not handled. This is only a deficiency of the current implementation, not of the algorithm as such.

4.2.8 Intersection singular-singular

Let x_i be an event x-coordinate such that $m_i^{(fg)} > 0$, $m_i^{(f)} \geq 2$, and $m_i^{(g)} \geq 2$.

First, we show that we only need to handle the case that we have a representation of x_i as a rational number. If $\deg(\gcd(R_{fgm_i^{(fg)}}, R_{fm_i^{(f)}}, R_{gm_i^{(g)}})) = 1$, we know $x_i \in \mathbb{Q}$. If the degree is larger, then there are common complex zeroes r_1, r_2 of all three resultants, singularities (r_1, s_1), (r_2, s_2) of f, singularities (r_1, s'_1), (r_2, s'_2) of g and intersections (r_1, s''_1), (r_2, s''_2) of f and g.

The conditions on $\{f, g\}$ require $s_j = s'_j = s''_j$ for $j \in \{1, 2\}$. Now recall from §3.3.2 that two distinct singularities on a cubic f have a common line component through them, so that $s_j = s'_j = s''_j$ contradicts the coprimality of f and g. Since coprimality has already been checked in §4.2.2, we may conclude that the noncoverticality condition has been violated if the gcd above does not yield a rational representation of x_i.

The representation of event x-coordinates as in §2.4.4 with its automatic simplification of polynomials means that $\gcd(R_{fgm_i^{(fg)}}, R_{fm_i^{(f)}}, R_{gm_i^{(g)}})$ has already been computed implicitly when the sequences of resultant zeroes were merged in **Two-curve Event x-Coordinates**.

With $x_i \in \mathbb{Q}$, factorization of $f(x_i, y), g(x_i, y) \in \mathbb{Q}[y]$ by multiplicities yields rational values for the y-coordinates of all points of f and g over x_i, making the analysis easy.

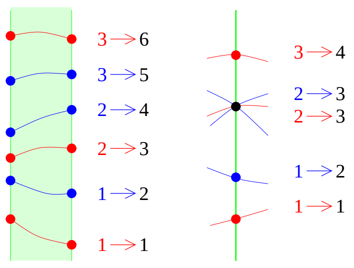
> **Intersection singular-singular:** Let $m_i^{(fg)} > 0$, $m_i^{(f)} \geq 2$, and $m_i^{(g)} \geq 2$.
>
> If x_i is not represented as a rational, signal "covertical events" and abort.
>
> Factor $f(x_i, y)$ and $g(x_i, y)$ by multiplicities to obtain the y-coordinates of all points over x_i and sort them.
>
> If the two singularities are different, or both curves have a common non-singular point over x_i, signal "covertical events" and abort. Otherwise, the geometry is now clear.

This concludes the case distinction for the analysis of two-curve events.

4.2.9 Slices of curve pairs

The analysis of $\{f, g\}$ produces information on the relative position of arcs and event points in y-direction in the form of arc sequences over intervals between events and below/above event points. For the benefit of arrangement computation (see Section 5.2), we recast this data in a uniform way that supports indexing by arc numbers w. r. t. an individual curve.

Let ξ be an x-coordinate, and let μ be the arc number of a point p on f over ξ (be it the arc number of an event point p, or be it the number of an arc containing p in its interior). We define the *slice* of the curve pair $\{f, g\}$ at ξ to be a pair of tables, one for f and one for g, such that the μ-th entry of the f-table contains the arc number of p on fg. In other words, the f-table maps μ to ν iff the μ-th point of f over ξ is the ν-th point of fg over ξ. Similar for g.



Obviously, there is only a finite number of different slices: One for each event x-coordinate and one for each interval between events. After locating an arbitrary real algebraic number ξ represented as in §2.4.4 within the sequence of two-curve event x-coordinates, it is straightforward to construct the slice of $\{f, g\}$ at ξ.

Arc numbers and slices are the closest analogue to y-coordinates provided by our analysis. We will see in §5.2.2 that they suffice for our purposes of comparsions in y-direction.

The slices over the first and last interval between events reflect the behaviour of $\{f,g\}$ before and after the finitely many two-curve events for $\xi \to \pm\infty$, respectively. Hence we adopt the convention of calling them the slices *at* $\pm\infty$. This does not reflect the behaviour of the curves at infinity in the sense of projective geometry in $\mathbb{P}^2(\mathbb{R})$ or of topology in the plane $(\mathbb{R} \cup \{\pm\infty\}) \times \mathbb{R}$ with a compactified x-axis. Instead, these are the slices at values of ξ that are *infimaximal*: standing for finite quantities of arbitrarily large magnitude, larger than any event x-coordinate or any other quantity considered by our analyses. (The term infimaximal seems to originate with [MS01].)

4.3 Choosing Coordinates

4.3.1 Characterization of forbidden coordinate systems

The algorithm for the analyses of one and two curves has imposed conditions on the choice of a coordinate system in §4.1.1 and §4.2.1, demanding

(i) nonverticality of tangents at flexes, singularities, and intersections,

(ii) y-regularity of curves, and

(iii) noncoverticality of event points, including complex intersection points and events on the Jacobi curve.

These conditions can be formulated as nonverticality of a finite number of lines:

(i) nonverticality of tangents at flexes, singularities, and intersections,

(ii) nonverticality of the factors of the highest-order terms (see §3.2.2), and

(iii) nonverticality of lines joining any two event points.

So it seems that we are ready to take this finite set L of lines and apply Proposition 3.2.1 to demonstrate that all but finitely many choices of a shearing parameter r yield a coordinate system in which the conditions are satisfied.

Unfortunately, this is not quite so simple, since the definition of L refers to x-extreme points, and they depend on the choice of a specific coordinate system: The lines joining all pairs of x-extreme points in the original coordinate system may be nonvertical after shearing, but they are no longer the lines through the x-extreme points! For the other event points there is no such problem, since flexes, singularities, and intersections (including those involving the Jacobi curve) are independent of the choice of coordinates. Let us call them the *invariant event points*.

The crucial property of a vertical line through an x-extreme point of a curve f is the fact that it is a tangent, and the tangency to f at this point is invariant under coordinate changes, whereas the property of being x-extreme is not. Hence the noncoverticality of some invariant event point v to all x-extreme points of f is guaranteed if all tangents to f containing v are nonvertical. Similarly, the noncoverticality of all x-extreme points of f to all x-extreme points of g is guaranteed if all lines tangent to both curves are nonvertical. (Recall that coverticality of two x-extreme points on one curve is excluded automatically by Proposition 3.3.1.)

It remains to prove that these critical tangents are finite in number. A powerful tool for doing so is the concept of a dual curve [BK86, pp. 251+] [G98, 16.6]. (We can only give an informal introduction here and refer the reader to the literature for more.) Recall the duality of points and lines in the projective plane $\mathbb{P}^2(K)$: A triple of coordinates, up to a common factor, can either be regarded as a point $(a : b : c)$ or a line equation $ax + by + cz$, and exchanging these roles preserves the incidence relation of lines and points, since $ax + by + cz = 0 \Leftrightarrow xa + yb + zc = 0$. Let $F \in \mathbb{C}[x,y,z]$ be a projective algebraic curve without line components. Then its *dual curve* is the set of all lines tangent to F (in whichever point), and it is in fact a curve, i. e. there exists a homogeneous polynomial $F^* \in \mathbb{C}[x,y,z]$ defining this point(!) set. F^* has no line components, and $F^{**} = F$.

Let $G \in \mathbb{C}[x,y,z]$ also be a projective algebraic curve without line components. We claim that $(FG)^* = F^*G^*$, because a line is a tangent to FG iff it is a tangent to F or a tangent to G. It follows that F and G are coprime iff F^* and G^* are coprime.

Proposition 4.3.1: *Let $F \in \mathbb{C}[x,y,z]$ be a projective algebraic curve, and let $v \notin F$ be a point. Then there are only finitely many tangents to F containing v.*

Proof: Let us first treat the case that F has no line components. Assume w. l. o. g. that $v = (0 : 0 : 1)$ (otherwise change coordinates). The set of all lines $ax + bx + cz$ containing v is the subset $\{(a : b : c) \mid c = 0\}$ of the dual plane, hence a line. By Bezout's Theorem, it intersects F^*, which is free of line components, in finitely many points, corresponding precisely to the tangents to F containing v.

Now if F has a line component H, the only tangent to H is H itself, which may or may not contain v, so that the full claim follows by induction on the number of line components. ☐

In fact, one can derive an explicit bound $n(n-1)$ on the number of tangents containing v as a function of the degree $n := \deg(F)$ in a more direct fashion, see [BK86, Prop. 5.2.2]. This is actually a bound on $\deg(F^*)$, see [BK86, Prop. 6.2.7].

Proposition 4.3.2: *Let $F, G \in \mathbb{C}[x,y,z]$ be two coprime projective algebraic curves. Then there are only finitely many lines tangent to both F and G.*

Proof: First assume that F and G have no line components. Their coprimality entails the coprimality of their duals F^* and G^*, so that Bezout's Theorem implies the finiteness of $F^* \cap G^*$, the set of common tangents. Adding a line component H to, say, F adds just H to the set of tangents to F, so that the full claim follows inductively. $\square$

These propositions prove that for each pair of coprime square-free algebraic curves $f, g \in \mathbb{Q}[x,y]$ there is a finite set L_{fg} of lines such that for all choices of a coordinate system, the nonverticality of all lines in L_{fg} guarantees that the conditions on the coordinate system imposed in §4.1.1 and §4.2.1 are satisfied. In fact, a careful study of the preceding arguments, taking into account that $\deg(f)$ and $\deg(g)$ are bounded by a constant, shows that L_{fg} can be chosen with a size bounded by some constant which is independent of the particular choice of f and g.

4.3.2 Random Shearing

The results of §4.3.1 in conjunction with Proposition 3.2.1 justify the following strategy to ensure the conditions imposed on $\{f,g\}$:

> **Random Shear:** Choose a random shearing parameter $r \in \mathbb{Q}$.
> Replace f,g by $f \circ S_{-r}$ and $g \circ S_{-r}$, respectively.
> Perform the analysis of $\{f,g\}$.
> If a violation of the conditions is signalled, restart with a new r.

It remains to define a strategy for choosing r. The traditional probabilistic way would be to bound the number of unlucky values of r and choose r randomly from a range of values $k \geq 2$ times as big, leading to a failure probability of k^{-1}. However, we expect most unlucky parameter values to lie outside that range: rational with large bit length, or irrational. So the traditional approach seems likely to overestimate the failure probability. This is harmful to the performance of the algorithm, since choosing from a large range means choosing a parameter r with large expected bit length s, and the bit length of the coefficients of f grows by up to $s\deg(f)$ bits during shearing.

Therefore, we suggest to start with a small parameter range and to have it grow exponentially in the number of past failures.

When trying to avoid rational in favour of integer arithmetic, we suggest not to pick integer $r \in \mathbb{Z}$, since shearing parameters with a large magnitude smash the scene and can make formerly innocuous scenes ill-conditioned by introducing very shallow intersections etc. (Of course our algorithm can handle this, but it consumes more time in e. g. the comparison of y-coordinates.) Instead, we suggest to simulate shearing with $r = s/t$, $s,t \in \mathbb{Z}$, $t \geq 1$ by a combination of scaling and shearing of the form

$$\begin{pmatrix} t & 0 \\ 0 & t \end{pmatrix} \begin{pmatrix} 1 & s/t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} t & s \\ 0 & t \end{pmatrix} \in \mathbb{Z}^{2\times 2} \tag{4.5}$$

and choosing t larger than $|s|$ (but not too large either, otherwise formerly covertical points will tend to have very close x-coordinates).

4.3.3 Shearing back the results

Suppose that analyzing the geometry of a curve or a curve pair has required to change coordinates with a shear S_r. Mathematically, it seems clear how to transport back the results into the original coordinate system: Just map them with S_{-r}. Unfortunately, this is not compatible with the "y per x" view taken for the analysis: For example, an intersection point of f and g at ξ found to lie on the μ-th arc of f and the ν-th arc of g is not represented in a way that allows us to evaluate S_{-r} on it, because an explicit y-coordinate is lacking. We do not want to introduce symbolic y-coordinates since we set off to do without them for performance reasons in the first place.

Instead, our algorithm defines a post-processing step in which a high-precision floating point approximation of event points is computed, obeying a user-defined error bound for the results: Compute the x-coordinate in the analysis' sheared coordinate system by refining the isolating interval to a sufficient precision. Then substitute it into the defining curve f and solve for y. If this would require solving for a multiple zero, use f_y instead of f. (The analysis of one curve identifies exactly those cases in advance, see Corollary 3.2.10.)

The mechanisms needed for this, especially the necessary backward analysis, are a central part of the forthcoming implementation of the `leda::real` diamond operator (see §2.4.3). Therefore, realizing this part in the implementation has been postponed and is not done in the version prepared for this thesis.

It needs to be stressed that there is no reliance on the distinctive capabilities of the diamond operator being incurred here, just a reuse of an item of numerical analysis forming part of its implementation.

For fast graphical output or similar purposes, the computation described above can be performed sufficiently well without error guarantees using `double` arithmetic. This has been implemented for the `xcubi` demo program.

In either case, the conversion from the implicit representation in the analysis' coordinate system to the numeric coordinates in the original coordinate system is a one-way street. As a consequence, an algorithm invoking the analyses of various pairs of curves, like the arrangement computation of the next chapter, needs to choose a sheared coordinate system globally.

As far as the purely topological information (such as the planar map representing an arrangement) is concerned, shearing back is not an issue, since the topology is invariant under orientation-preserving coordinate changes.

4.3.4 Design Rationale

We faced the fundamental question of whether to impose conditions requiring coordinate changes—no matter how seldom—or not.

The main argument against is the problem of shearing back just discussed in §4.3.3. We consider our algorithm as a contribution to the domain of exact computational geometry, not symbolic algebra. Therefore the restriction to numeric but arbitrarily precise results seems tolerable, following the idea that "the exactness is in the geometry, not in the arithmetic" [YM01, 3.].

The main argument in favour is keeping the algorithm to a more manageable size: more manageable in terms of development, implementation, credible test coverage, and maintenance. It is hard to identify a single degeneracy that absolutely would have escaped treatment by adding yet another few subcases. But in summary the amount of removed special cases is considerable: spurious zeroes of resultants, vertical asymptotes, case distinctions for arcs not running from left to right but taking turns at event points, vertical line components, covertical intersections with hard-to-determine multiplicities, etc.

After the fundamental decision in favour of shearing has been taken, one has to cope with the burden of shearing back anyway. Therefore, it appears coherent to make liberal use of this facility once it exists, as long as the number of forbidden y-axes per curve pair remains bounded by a constant. Therefore we give no detailed defence here of each condition imposed, only of the general principle.

An alternative to shearing with a concrete value r is to keep this value symbolic. Algebraic operations are then done over $\mathbb{Q}(r)$ instead of $\mathbb{Q}$, and three-valued comparisons in $\mathbb{Q}(r)$ are performed by regarding r as an infinitesimally small positive number. In a straight-line setting, a similar approach [MS01] is known to work well (for a symbolic quantity interpreted as tending to $+\infty$). In our setting, however, this approach would perform much worse due to the expensive symbolic computations like resultants and gcds. Therefore, we did not follow this idea.

Chapter 5

Arrangements of Segments of Algebraic Curves

5.1 Arrangement Computation

5.1.1 Segments and their Arrangements

Let $f \in \mathbb{R}[x,y]$ be an algebraic curve. A *segment* s of f is a subset $s \subseteq f$ which is either a single point (then we call it a *trivial segment*), or a connected closed subset of f whose interior is a C^∞-manifold of dimension 1 and homeomorphic to an open interval. Thus we can say that a non-trivial segment is a "smooth piece of curve between two points (or extending to infinity)". A prominent example of segments are arcs as introduced in §3.2.5; but not all segments are arcs: A segment may contain an x-extreme point or pass smoothly through a singularity along a real branch, and a segment may end independently of one-curve events. If a subset $s' \subseteq s$ of a segment s is also a segment, it is called a *subsegment* of s.

In analogy to arcs, we call the boundary points of a segment its *endpoints*. A bounded non-trivial segment has exactly two boundary points. By sorting them lexicographically, we may speak of the (smaller) *source point* and the (larger) *target point* of s. For a trivial segment $s = \{v\}$, we call v both the source and the target point of s. For an unbounded segment, we get a missing source or target point in the obvious fashion by talking of endpoints *at infinity*. With these conventions, every segment has two endpoints (maybe identical or at infinity). The implementation also allows *reversed* segments with source larger than target.

A finite set of segments induces a subdivision of the plane into vertices, edges, and faces, called the *arrangement* of the segments. Its vertices are the endpoints and intersection points of the given segments. Its edges are the interiors of the minimal subsegments of the given segments that connect adjacent vertices. Its faces are the connected open subsets of the plane bounded by the union of vertices and edges. The topology of an arrangement can be represented by a planar map [MN99, 8.1–8.6], that is a directed graph which is bidirected (every edge possesses a reversal) and annotated with the cyclic ordering of the edges leaving each vertex. The geometric data—point coordinates for the vertices and segments supporting the edges—can be attached to the vertices and edges, respectively.

Our goal is to compute the arrangement induced by a set of segments of algebraic curves of degree up to 3. To this end, we consider the restricted problem of arrangement computation for segments subject to the following conditions:

- All curves satisfy the requirements of §4.1.1.
- All curve pairs inspected by the algorithm satisfy the requirements of §4.2.1.
- Every segment is a subset of an arc of the supporting curve or is an isolated point; and all points in the interior of the segment have the same arc number.

Segments fulfilling the last condition are called *sweepable*. The necessity of this condition will become clear in the sequel. We return to general segments in Section 5.3. From now on until then, the term "segment" denotes a sweepable segment.

The reader is invited to observe that the restriction to degree ≤ 3 is not inherent to the approach taken in this chapter, it is just a restriction of the underlying analyses of curves and curve pairs from Chapter 4.

A general paradigm for designing geometric algorithms in the plane is the use of a *sweep line*: enumerate and process the elements of the scene in that order in which a vertical line swept over the plane from left to right hits them. The well-known algorithm by Bentley and Ottmann for computing the arrangement of straight-line segments in the plane [BO79] is based on this idea. In its original form, this algorithm cannot handle degeneracies like intersections of more than two segments or intersections coinciding with endpoints. We base our work on a refined version due to Mehlhorn and Näher [MN94] [MN99, 10.7] which can handle all situations. In particular, segments may overlap, and an arbitrary number of segments may start, end, and intersect in a single event point. Two segments are said to *overlap* if their intersection contains an open set (i. e. is infinite).

Already in their original publication, Bentley and Ottmann have observed that their algorithm can be generalized to curved segments, as long as the necessary predicates and constructions are provided for the given segments. A *predicate* is a function on a fixed number of geometric objects that has a small finite range of values, typically $\{$true, false$\}$ or $\{$less, equal, greater$\}$. (Examples: "Does point v lie on segment s?", "Does point v lie below, on, or above segment s?") A *construction* takes a fixed number of geometric objects and maps them to a new geometric object. (Example: computing the intersection point of two intersecting lines.)

In the rest of this section, we will give an outline of the Bentley-Ottmann algorithm and discuss which predicates and constructions are necessary.

5.1.2 The generalized Bentley-Ottmann algorithm

To compute a planar map representing an arrangement of segments, sweep a vertical line over them and preserve the following invariant: Left of this *sweep line*, the planar map has already been constructed. The segments intersecting the sweep line at its current position are stored in a sequence called the *Y-structure*. It is sorted in ascending y-order of intersection points. Right of the sweep line, all segment endpoints and some intersection points—at least those of segments being adjacent in the Y-structure—are stored in a queue called the *X-structure*. The X-structure is sorted in lexicographic order.

Intersections and endpoints of segments are collectively called *event points*, because it is only at these points that the status of the sweep line changes. A segment containing an event point is said to be *involved* in the event. The conceptual sweep over the whole plane amounts to advancing the sweep line over this finite number of points. Mehlhorn's and Näher's careful formulation of the algorithm uses an infinitesimally tilted sweep line which sweeps over covertical event points in the order of their y-coordinates; hence the lexicographic sorting of events.

The algorithm performs the following steps until the X-structure has been emptied:

1. Extract next event point from the X-structure. Find segments involved in the event by locating the event point in the Y-structure, exploiting its order.
   (Requires: Test if the event point is above, on, or below a given segment.)

2. Remove ending segments, i. e. those with target point equal to event point.
   (Requires: Comparison of event point and target points.)

3. Reorder remaining intersecting segments according to multiplicity of intersection (explained below in §5.1.3) such that the new order reflects the situation right of the event.
   (Requires: Test if segments overlap. — Intersection multiplicity of non-overlapping segments.)

4. Add starting segments to the Y-structure according to its ordering.
   (Requires: Comparison of event point and source points. — Comparison of segment y-order right of a common point.)

5. Add intersections of newly adjacent segments to the X-structure,[1] obeying its ordering.
   (Requires: Computing intersection points. — Lexicographic comparison of points.)

[1]This is the crucial point of the sweep line algorithm: Only adjacent segments have their intersections computed, avoiding the naive $O(N^2)$ analysis of all pairs of segments.

Observe how the use of predicates by the algorithm automatically reduces all geometric analyses to at most two curves at a time, even if many segments run through one event point.[2]

In summary, we need the following predicates and constructions:

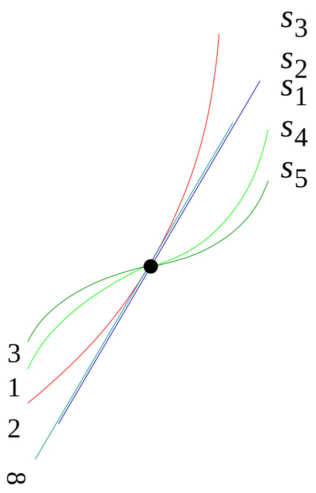
- Lexicographic comparison of intersections and endpoints of segments.
- Comparison of y-coordinates of a point and a segment.
- Comparison of segments in y-direction right of a common point.
- Segment overlap test.
- Computation of segment intersection points.
- Intersection multiplicity of non-overlapping segments.

We present their realizations in Section 5.2.

Now one can see why we required segments to be subsets of arcs: Firstly, this excludes x-extreme points, making segments *x-monotone* (every homeomorphic parametrization is strictly monotone in its x-coordinate). This guarantees that the source point of a segment is indeed the first point hit by the sweep line. Secondly, this excludes singularities, which makes sure that the intersection multiplicities of the supporting curves properly reflect the behaviour of the segments (see below).

5.1.3 Reordering segments passing through an event

The reordering step 3. of the Bentley-Ottmann algorithm requires further explanation, especially since this is the one place where the treatment of curved segments differs from the straight-line case on this level of abstraction. What we present here comes from [BE+02].



In the reordering step, only segments containing the event point in their interior are involved. Let us call them $s_1, \dots, s_k$, numbered in ascending y-order just left of the event. Since the s_i are sweepable segments, the intersection occurs in the interior of arcs of the respective supporting curves, and we can write a segment s_i locally as an analytic implicit function

$$y = \varphi_i(x) = \sum_{d=1}^{\infty} a_d^{(i)} x^d \tag{5.1}$$

after translating the event point to $(0,0)$, see §3.2.5.

[2]Our representation of a point involves only one curve (see §5.2.1) so that this view is justified, even though, for example, the comparison of two intersection points can be seen as involving four curves, viz. the two pairs of curves creating the intersections.

If two such segments overlap, they coincide with each other on both sides of the event, and we define their intersection multiplicity as ∞. Otherwise, there is an intersection of finite multiplicity (in the sense of §3.2.3), and we are in the situation of Proposition 3.2.11. In either case, the intersection multiplicity of s_i and s_j is $\min(\{d \mid a_d^{(i)} \neq a_d^{(j)}\} \cup \{\infty\})$.

The coefficients of the implicit functions determine the y-order of segments just left and just right of the intersection.

Proposition 5.1.1: *With notation as above:*
Segment s_i lies below segment s_j right of the intersection iff

$$(a_1^{(i)}, a_2^{(i)}, \dots, a_d^{(i)}, \dots) <_{\text{lex}} (a_1^{(j)}, a_2^{(j)}, \dots, a_d^{(j)}, \dots).$$

Segment s_i lies below segment s_j left of the intersection iff

$$(-a_1^{(i)}, a_2^{(i)}, \dots, (-1)^d a_d^{(i)}, \dots) <_{\text{lex}} (-a_1^{(j)}, a_2^{(j)}, \dots, (-1)^d a_d^{(j)}, \dots).$$

(Here $<_{\text{lex}}$ is the lexicographic order relation on sequences of real numbers.)

Proof: It suffices to demonstrate the first part; the second part follows by substituting $-x$ for x. Iff the segments overlap, they coincide around the intersection and have equal coefficient sequences. Otherwise, $m := \min\{d \mid a_d^{(i)} \neq a_d^{(j)}\} < \infty$. Then $\varphi_i(x) - \varphi_j(x) = (a_m^{(i)} - a_m^{(j)})x^m + \text{HOT}$ is negative for small $x > 0$ iff $a_m^{(i)} < a_m^{(j)}$. $\square$

In general, intersection multiplicities will not have been computed for all $\frac{1}{2}k(k-1)$ pairs of segments when we come to process their common intersection. For $1 \leq i < k$ let $m_i \in \{1, 2, 3, \dots, \infty\}$ be the intersection multiplicity of the adjacent segments s_i and s_{i+1}. These multiplicities have already been computed when s_i and s_{i+1} were found to overlap or intersect, resp., in the current event (see Section 5.2). The following result allows us to infer all other intersection multiplicities from them.

Proposition 5.1.2: *With notation as above, the intersection multiplicity of s_i and s_j, $1 \leq i < j \leq k$, is* $\min\{m_i, \dots, m_{j-1}\}$.

Proof: The proof rests on the interpretation of intersection multiplicity as number of agreeing Taylor coefficients (Proposition 3.2.11) and proceeds by induction on j. The base case $j = i + 1$ is clear from the definition of m_i.

For the inductive step from j to $j + 1$, let $m = \min\{m_i, \dots, m_{j-1}\}$ be the intersection multiplicity of s_i and s_j. For $m_j = \infty$, the claim is clear. Otherwise, distinguish three cases:

If $m > m_j$, then
$$\begin{aligned} a_d^{(j+1)} &= a_d^{(j)} = a_d^{(i)} && \text{for } d < m_j, \\ a_d^{(j+1)} &\neq a_d^{(j)} = a_d^{(i)} && \text{for } d = m_j, \end{aligned}$$

so that the intersection multiplicity of s_i and s_{j+1} is $m_j = \min\{m, m_j\}$.

If $m < m_j$, we have equality for $d < m$ as above and inequality $a_d^{(j+1)} = a_d^{(j)} \neq a_d^{(i)}$ for $d = m$, demonstrating the intersection multiplicity $m = \min\{m, m_j\}$.

However, if $m = m_j$, then only a double inequality $a_d^{(j+1)} \neq a_d^{(j)} \neq a_d^{(i)}$ holds for $d = m$, but we need $a_m^{(j+1)} \neq a_m^{(i)}$. Proposition 5.1.1 helps: Since s_{j+1} lies above s_j and intersects with multiplicity $m_j = m$, we know $(-1)^m a_m^{(j+1)} > (-1)^m a_m^{(j)}$. By the analogous argument for s_j and s_i, we know $(-1)^m a_m^{(j)} > (-1)^m a_m^{(i)}$. Hence $(-1)^m a_m^{(j+1)} > (-1)^m a_m^{(i)}$, as required. □

The proposition justifies the following method of performing step 3. (see [BE$^+$02]):

> **Reorder Segments:** Let $s_1, \ldots, s_k$ be segments passing through a common event point, numbered in ascending y-order left of the event point.
> Let $m_1, \ldots, m_{k-1}$ be the intersection multiplicities of adjacent segments.
>
> Let M be an even upper bound of all finite m_i.
>
> For $m = M, M-1, \ldots, 1$, take all maximal subsequences $s_i, s_{i+1}, \ldots, s_j$ with the property $m_i, m_{i+i}, \ldots, m_{j-1} \geq m$ and reverse their order.

Observe that the intersection multiplicities m_i are not readjusted to reflect the intersection multiplicity of s_i and s_{i+1} once the segments are being moved.

To prove correctness, let us first consider two non-overlapping segments, initially numbered s_i and s_j, which have been put into the same subsequence exactly n times. This is equivalent to $n = \min\{m_i, \ldots, m_{j-1}\}$, since they were first put together for $m = n$ and then again in every subsequent iteration. The segments have changed their relative position iff n is odd. By Proposition 5.1.2, n is their intersection multiplicity. By Corollary 3.2.12, the segments have to change their relative position iff n is odd. Hence they were rearranged correctly. Let us now consider the special case of two overlapping segments s_i and s_j. They belong to a sequence of pairwise overlapping segments $s_i, s_{i+1}, \ldots, s_j$, so that $m_i = m_{i+1} = \ldots = m_{j-1} = \infty$, which implies that s_i, s_j have been put together M times and have not changed their relative position (because M is even), which is also correct.

A reader still surprised by the fact that there is no need to reorder the m_i in accordance with the permutation of segments may want to verify that the following invariant holds at the beginning of a loop iteration: If $m_i \leq m$, then it is the actual intersection multiplicity of s_i and s_{i+1}. If $m_i > m$, then the actual intersection multiplicity is also $> m$.

The time complexity of **Reorder Segments** is $O(M \cdot k)$. As an improvement, Lutz Kettner[3] suggested an $O(k)$ algorithm based on a linear-size tree that reflects the nesting of subsequences defined by minimum intersection multiplicities.

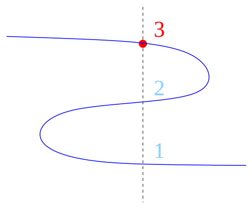
[3]Personal communication; publication of this result is pending.

5.2 Sweepable Segments and Points

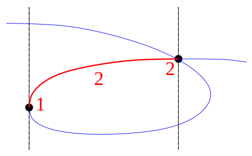
5.2.1 Representation of Points and Sweepable Segments

Now we define representations of sweepable segments and points on them that follow the approach of Chapter 4 so that predicates can be evaluated based on the analyses of curves and curve pairs described there. In other words, we finally make the link between chapters 4 and 5.

Let $v \in \mathbb{R}^2$ be a point with algebraic coordinates lying on an algebraic curve $f \in \mathbb{Q}[x,y]$. Let the x-coordinate of v be represented in the way of §2.4.4 as ξ, and let the arc number of v be μ (see Section 4.1). The triple (f,ξ,μ) represents v. In addition, we allow representations with the special value $\xi = \pm\infty$, standing for points of f at infinity (in the infimaximal sense of §4.2.9). For them, μ has to be an arc number that exists over the first/last interval between events of f.



Let $s \subseteq \mathbb{R}^2$ be a (sweepable) segment of an algebraic curve $f \in \mathbb{Q}[x,y]$. Let μ be the common arc number of all points in its interior. Since we allow points at infinity, s always has a source point v_- and a target point v_+. Let μ_- and μ_+ be the arc numbers of source and target point on f. We represent s by the sextuple $(f, v_-, v_+, \mu_-, \mu, \mu_+)$.



The x-*range* of a segment s is its pointwise projection to x-coordinates, that is the set of x-coordinates of all its points. The x-range is a closed interval bounded by the x-coordinates of source and target point.

Our decision to require a fixed arc number for all of a segment's interior was motivated by the intention to keep the representation of a segment simple and constant in size.

5.2.2 Comparison Predicates

The pivotal operations in our comparison predicates are comparisons of coordinates. All x-coordinates are represented in the way of §2.4.4 as defining polynomial plus isolating interval, allowing us to compare them straight away (see ibid.); hence let us turn to y-coordinates.

The y-coordinate of a point v needs no further explanation. The y-coordinate of a segment s for a given x-coordinate ξ in the x-range of s is the y-coordinate of the unique point on s over ξ.

For our purposes, y-coordinates of points and/or segments are only ever compared over a common x-coordinate ξ. (This motivates the "y per x" view taken in Chapter 4.) Their comparison amounts to comparing arc numbers: arc numbers w. r. t. a single curve f if both objects are supported by f, in which case the relevant arc numbers can be read off straight away from the objects; or arc numbers w. r. t. a curve pair $\{f,g\}$ if one object is supported by f and the other by a different curve g, in which case the arc numbers given in the objects can be translated to arc numbers w. r. t. the pair using the slice of $\{f,g\}$ at ξ (see §4.2.9).

Lexicographic comparison of points

Suppose we are given two points (f_1,ξ_1,μ_1) and (f_2,ξ_2,μ_2). If $\xi_1 \neq \xi_2$, the relation of x-coordinates determines their lexicographic order. If $\xi_1 = \xi_2 =: \xi$, compare their y-coordinates over ξ by comparing the appropriate arc numbers.

***y*-Comparison of point and segment**

Suppose we are given a segment $s = (f, v_-, v_+, \mu_-, \mu, \mu_+)$ and a point $v = (g, \xi, \nu)$ such that ξ lies in the x-range of s. The y-order of s and v is determined by comparing v lexicographically to an auxiliary point (f,ξ,μ') with an arc number $\mu' = \mu$ if ξ lies in the interior of the x-range or $\mu' = \mu_\pm$ (as appropriate) if ξ lies at the boundary of the x-range of s.

***y*-Comparison of segments right of a common point**

Suppose we are given two segments s_1, s_2 and a common point $v \in s_1 \cap s_2$. We want to compare the y-order of s_1 and s_2 just right of v. Comparing two segments supported by the same curve f amounts to comparing their interior arc numbers μ.

Comparing two segments supported by two different curves f and g amounts to comparing arc numbers with respect to $\{f,g\}$: Let ξ be the x-coordinate of the common point v. Locate ξ among the event coordinates of $\{f,g\}$, take the interval I to its right, and apply the slice of $\{f,g\}$ over I to the segments' interior arc numbers in order to obtain the relevant two-curve arc numbers.

5.2.3 Intersection Construction and Related Predicates

Handling intersections first requires to address overlapping, since two overlapping segments neither have discrete intersection points nor an intersection multiplicity in the sense of §3.2.3. Hence we forbid the evaluation of the intersection point construction and the intersection multiplicity predicate on pairs of segments that overlap.

Segment overlap test

As we require different curves to be coprime, which is checked when the resultant of the corresponding curve pair is computed, segments can overlap only if their supporting curves are the same. Two segments of the same curve overlap iff they are non-trivial, their interior arc numbers are the same, and their x-ranges overlap. These criteria are easy to check.

Segment intersection points

Two non-overlapping segments on one curve f can intersect only in their endpoints, so intersection computation reduces to comparing endpoints in this case.

Two segments on two different curves f and g can intersect only if their x-ranges have a non-empty intersection I. If this is the case, inspect for every two-curve event reported by the analysis of $\{f,g\}$ with an x-coordinate $\xi \in I$ whether it is an intersection and whether the involved arc numbers equal the arc numbers of the segments at ξ. This yields exactly the intersection points of the two segments.

Intersection multiplicity

Our conditions on segments and their supporting curves require different curves to be coprime and intersections in a curve pair to be noncovertical. Therefore, the multiplicity of an intersection point of two curves f and g at ξ is simply the multiplicity of ξ as a zero of $\operatorname{res}(f,g,y)$, which is computed by the analysis of a curve pair as part of the data on the intersection.

5.3 Input Segments and Curves

5.3.1 General Remarks

The sweep line method for arrangement computation described above applies to the restriced class of sweepable segments, as defined in §5.1.1. We are now going to lift these restrictions. To do so, we need to surpass two limitations inherent in the representation of sweepable segments:

1. The predicates for the Bentley-Ottmann algorithm are based on the analyses of Chapter 4. This means that the conditions of §4.1.1 and §4.2.1 limit the choice of a coordinate system for the segments during arrangement computation. The randomized technique of Section 4.3 for finding a suitable coordinate system is applicable, but requires a means of transforming the input segments. The representation of sweepable segments does not allow this.

2. A restriction to x-monotone segments is not acceptable. Already in itself, this is an unnatural restriction from an application point of view, but the problem is aggravated further by the fact that *any* segment, if not a horizontal line segment, may acquire an x-extreme point when sheared with an unlucky

shearing parameter, so that every interesting set of segments is in danger of becoming non-sweepable under a randomized shear.

Consequently, we define separate representations of (sets of) input segments that are general enough to cover interesting classes of segments, that are closed under applications of a shear S_r, $r \in \mathbb{Q}$, and that allow conversions to sets of sweepable segments. In accordance with our intent to avoid arithmetic with algebraic numbers in favour of rational numbers for the sake of efficiency, we restrict ourselves to input data whose numerical parts, in particular coordinates of endpoints, are rational.

It has to be stressed that this is not a restriction of the actual curve analysis and arrangement computation: As long as the supporting curves have rational coefficients (as required by the algorithms of Chapter 4), the endpoints of sweepable segments can be algebraic of arbitrary degrees (since the x-coordinates are represented with defining polynomials of arbitrary degrees). It is just a matter of shearing this data and transforming it to sweepable segments.

5.3.2 Input of Entire Curves

Let us first consider a simple but important special case of specifying segments. (Only this case has been implemented so far.)

An algebraic curve $f \in \mathbb{Q}[x,y]$ (after a change of coordinates of to make it y-regular, if necessary) can be regarded as a finite set of segments, viz. its arcs and isolated points. This allows us to subsume the problem of computing the arrangement induced by a set of curves under the heading of arrangement computation for segments.

Ensuring the required squarefreeness of curves has been discussed in §4.1.1. Pairwise coprimality can be achieved as follows: If $h := \gcd(f,g)$ is nonconstant for two curves f and g, replace them by the three curves $f/h, g/h, h \in \mathbb{Q}[x,y]$. To avoid gcd computations for all pairs of curves, one can defer it until the geometric analyses invoked by the predicates signal a pair with a common factor, and restart.

Shearing a curve f with S_r means to compute $S_r(f) = f \circ S_{-r}$ (see §3.2.2).

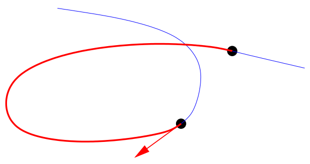
Transforming a curve into sweepable segments, based on its analysis with the method of Section 4.1, is not complicated either.

Curve to Segments: Perform the analysis of the curve f.
For all intervals I between events, and all parts $B = A \cap I \times \mathbb{R} \neq \{\}$ of arcs A over I, create a sweepable segment.
For all events that are acnodes, create a trivial sweepable segment.

The arc numbers $\mu_\pm$ at the boundaries can be inferred from the range of arcs involved in the event on either side of the interval I.

5.3.3 Input of Segments

Let us now consider the more typical case of individual input segments. An *input segment* is represented by a supporting curve f, endpoints $a, b \in \mathbb{Q}^2$, and a tangent vector $u \in \mathbb{Q}^2$ of f at a that points in the direction in which the segment s leaves a. For trivial segments, let $u = 0$. Only those values of (f, a, b, u) are allowed that actually describe a segment s.



With the results from Section 3.4, it is clear how to convert a cubic spline γ of the form (3.22) into this representation of an input segment: Check that the control points are not collinear (to avoid the case of a triple line) and take $\big(\mathrm{res}(P,Q,t), \gamma(l), \gamma(r), \gamma'(l)\big)$ with P and Q as defined there.

The representation of input segments allows shearing in a straightforward fashion: $S_r((f,a,b,u)) = (f \circ S_{-r}, S_r(a), S_r(b), S_r(u))$.

Specifying the tangent vector is necessary, for example, if a and b lie on an oval-shaped component of f such that there are two two segments linking them, or if a is a singularity such that a segment connecting a and b could leave a on more than one branch. For general algebraic curves, this could still leave ambiguities when a is a singularity, but for the restricted class of curves f considered here, where $\deg(f) \leq 3$, this data uniquely determines a segment. For trivial segments, this is trivial to see. For non-trivial segments s, it can be seen from the following method to trace the segment s on f, which also sketches an algorithm to break s into a set of sweepable segments.

To make this tracing process effective, use the analysis of f from Section 4.1. Translating the y-coordinates a_2 and b_2 to arc numbers amounts to locating them among the zeroes of $f(a_1,y), f(b_1,y) \in \mathbb{Q}[y]$, respectively.

- If f has no singularities (except maybe isolated points), its only event points contained in s are x-extreme points which link exactly two arcs. Therefore, one can trace s by starting in a and following the arcs of f through all event points. The only question is how to start from a: If $u_1 \neq 0$, its sign determines whether to start to the left or to the right. If $u_1 = 0$, then a is an x-extreme point, and $\mathrm{sign}(u_2)$ determines whether to leave the x-extreme point by the lower or the upper arc involved.

- When f has crunodes, tracing through them is simple, because the segment s is smooth so that it follows one of the two intersecting branches. Starting in a crunode a requires to compare u to the tangents of f at a.

- If f is an irreducible cubic with a cusp, tracing through the cusp is not an issue, since there is no smooth way through it. Starting in the cusp a cannot use u, since there is only one double tangent t in a. However, $\mathrm{mult}(a; f, t) =$

$3 = \deg(f)$ by Proposition 3.2.6, so that t has no further intersections with f according to Bezout's Theorem and partitions $\mathbb{R}^2$ into two halfplanes. Hence the arc of f to follow out of a is the one on the same side of t as b.

- If f consists of an irreducible conic h plus a tangent g and thus exhibits a tacnode, tracing through the tacnode works by exploiting the smoothness of s and remaining below or above the other branch. Starting in a tacnode a works by checking whether the unique tangent direction u is collinear to $b - a$. If so, s lies on the line component g, and we start tracing in b with a tangent vector of $-u$. Otherwise, s lies on the conic component h, which intersects g in no other point than a, and we start out of a along the h-branch. The h-branch is discernible as the lower (or upper) branch out of a if b is below (or above, resp.) the tangent at a.

Above, we have exploited the fact that a segment cannot "change component". We can even determine the component of f containing s as an element of $\mathbb{Q}[x,y]$ in the following way: If f consists of line and conic, we have seen how to factor f in §3.3.3. If f consists of line components, the line containing the rational points a and b has a rational equation obvious from a and b.

Besides lowering degrees, this is one way to achieve the required coprimality of different supporting curves f and g. The other way is to catch errors signalled by the geometric analyses invoked by the predicates, compute $h = \gcd(f,g)$, redefine the offending segments using one of f/h, g/h, or h, and restart.

5.4 Further Applications

The analyses of Chapter 4 and geometric predicates based on them like those of Section 5.2 have applications beyond the Bentley-Ottmann algorithm. Essentially the same set of predicates allows regularized boolean operations on polygons bounded by such segments [BE$^+$02] [MN99, 10.8]. This has not been considered in this thesis due to time restrictions; however, we do not expect any fundamental obstacles in this direction.

Also, other arrangement computation algorithms than Bentley-Ottmann can be supplied with predicates on the same basis, provided they are compatible with the "y per x" view of the geometric analyses and in particular do not require the comparison of y-coordinates of points over distinct x-coordinates. In particular, a randomized incremental construction could be used. It does not have to sort all event points and may thus reduce the number of expensive comparisons of event x-coordinates.

Chapter 6

Implementation

This chapter reports on the implementation of our algorithm as part of the EXACUS library and on running time measurements.

6.1 Overview

EXACUS is a collection of C++ libraries for efficient and exact algorithms for curves and surfaces that is being developed at the Max-Planck-Institut für Informatik as part of the European Union's Effective Computational Geometry project (ECG). Its overall design, due to Lutz Kettner and Susan Hert, is based on the generic programming paradigm [A98] and follows some ideas pioneered by CGAL [HH$^+$01].

The parts of EXACUS relevant here are the following:

- The **Library Support** (LiS) offers basic services such as a generic `Handle` template, I/O formatting conventions, support functions for iterators and the like.

- The **NumeriX** (NiX) library hosts polynomials, algebraic numbers, matrices, etc. In particular, the algebraic operations discussed in Chapter 2 are provided there. Parts of NumeriX were written or extended by the author during the preparation of this thesis or in his preceding programming practical.

- The **CubiX** (CbX) library is the main contribution of this thesis to EXACUS. Its two pivotal parts are the classes `Cubic_curve_2` and `Cubic_pair_2` which implement the curve and curve pair analyses of Chapter 4.

- The **SweepX** (SoX) library offers the LEDA-descendant sweep of *X* for arrangement computation, as described in Chapter 5, where *X* is an arbitrary class of segments that offers the necessary predicates and constructions. SweepX also provides LEDA-descendant regularized boolean operations on polygons bounded by such segments.

  The author has added a module for generic algebraic points and segments (GAPS) to SweepX, implementing the ideas of Section 5.2 in the form of class templates `Algebraic_point_2` and `Algebraic_segment_2` that are parametrized by curve pair analyses.

All parts of EXACUS have reference documentation for the library user embedded in the source code, which can be extracted and formatted using `doxygen` [Dox]. The abstract knowledge of our algorithms from the preceding chapters together with the overview given above should enable the reader to easily find any relevant information in the HTML version of the manual, starting from `doc.html` in the top-level EXACUS directory.

A more casual presentation is given in the next section, including a discussion of implementation details.

The omissions in the current version of the implementation are those indicated in the description of the algorithm: Intersections of two curves involving a regular point and a triangle's irrational acnode (§4.2.7); shearing back with arbitrary and guaranteed precision (§4.3.3); input of input segments instead of entire curves (§5.3.3).

6.2 A guided tour through the code

6.2.1 Welcome

Welcome to the EXACUS root directory `EXACUS`. (It is available to the thesis examiners and is going to become available to everyone through a public EXACUS release at some point in the future.) You will find that it contains various files related to the configuration and compilation of the EXACUS software for your system. How to do this is detailed in `EXACUS/Developers_manual/manual.ps`.

More importantly, there are subdirectories for all EXACUS libraries, in particular `EXACUS/CubiX` and `EXACUS/SweepX`. Since almost all of the library source code consists of templates and hence resides in header files, most of it can be found in the directories `EXACUS/`*library-name*`/include/`*library-abbrev*`/`. For example, the CubiX code is located in `EXACUS/CubiX/include/CbX/`. Typically, header files are named after the class template whose definition they contain.

Most major classes are implemented as handles and reps. That is, an object of class C is just a smart reference-counted pointer to a representation object of class C`_rep` that contains the actual data. This allows classes with large underlying representations to be copied around freely, with no memory-management effort required on the user's side. In LEDA parlance, they are *independent item types* [MN99, 2.2].

6.2.2 CubiX, part one

Analysis of a curve

The analysis of a cubic curve (Section 4.1) is implemented by the class template `Cubic_curve_2`. Like all of EXACUS, it is not tied a priori to a specific set of number types. We use it with the LEDA integer, rational, and `leda::real` number types together with matching instantiations of NumeriX's polynomials, algebraic numbers, etc. All these are collected in the `LEDA_arithmetic_traits` (defined in `NiX/Arithmetic_traits.h`) which we supply as the template parameter of `Cubic_curve_2`. Alternatively, the GMP/CORE set of number types could be used.

This is a typical use of a *traits class*: A set of related types is collected under standard names to serve as a kind of "compound" template argument for classes and functions which are written in a *generic* way, i. e. independent of a specific choice of these underlying types.

An object of class `Cubic_curve_2` is constructed from a bivariate polynomial defining the curve. We require integer coefficients for the curve. This is not a restriction and naturally leads to integer coefficients for all resultants of curves and all factors of resultants, including the defining polynomials of event x-coordinates represented in the style of §2.4.4; cf. the remark in §4.1.1.

The geometric analysis of a curve is performed by invoking the respective member functions of the curve object. Their names are not those of Chapter 4 like k_{i+1}, but more self-explanatory names like `.arcs_over_interval(i)`. Note the index shift: Almost all indices in the implementation are zero-based, to match the C++ array indexing convention.

The information provided by `Cubic_curve_2` includes the following:

- `.f()` – the defining polynomial f
- `.resultant_f_fy()` – the resultant $R_f = \mathrm{res}(f, f_y, y)$
- `.num_events()` – number of one-curve events
- `.event_x(i)` – i-th event x-coordinate
- `.event_info(i)` – information on i-th event (kind of event, involved arcs)
- ...

Information is computed on demand only. For example, R_f is not computed until a member function is called that needs it (like `.num_events()`). Furthermore, information is cached. So any subsequent call to `.num_events()`, `.resultant_f_fy()` and the like is almost for free. In particular, the analysis of each event point is delayed until the respective event is actually queried.

The `Event1_info` structure computed for each one-curve event looks like this (including `doxygen` comments):

```
struct Event1_info {
    Event1_kind kind;  //!< kind of event
    //! on LEFT and RIGHT side: range of arc numbers involved in event
    Interval<int> arcs[2];
    int numarcs_at;    //!< number of distinct arcs at event x-coord
    int arc_at;        //!< arc number of event point
};
```

For historical reasons[1], the actual code to analyze event points is contained in the separate file `CbX/compute_event1_info.h`. This fact is invisible for the library user.

Analysis of a curve pair

The class template `Cubic_pair_2` implements the analysis of a curve pair (Section 4.2). Its template parameter is an instance of `Cubic_curve_2`. A curve pair object is constructed from two curve objects, referred to as curves 1 and 2, or more concisely as f and g.

`Cubic_pair_2` provides member functions like the following:

- `.curve(i)` – the underlying curve 1 or 2
- `.resultant_f_g()` – the resultant $R_{fg} = \text{res}(f, g, y)$
- `.num_events()` – number of two-curve events
- `.event_x(j)` – j-th event x-coordinate
- `.event_info(j)` – information on the j-th event
- `.slice_at(x)` – slice of $\{f, g\}$ at x
- ...

Recall that two-curve events comprise intersections of f and g, as well as one-curve events on either of them. So the sequence of two-curve event x-coordinates consists of the real zeroes of R_f, R_g, and R_{fg} (§4.2.2), where some zeroes may coincide.

[1]At one point it seemed promising to decouple the cubic curve object that enforces degree ≤ 3 from the geometric analyses which might also be applicable to objects of other classes that could also represent cubic curves sometimes. This was aimed at code reuse in the context of Eric Berberich's work on quadrics.

There is a further member function `.event_indices(j)` which maps the index j of a two-curve event to the triple of corresponding indices w. r. t. each of the three resultants, with -1 denoting the non-existence of a corresponding index.

The ideas of lazy evaluation and caching are followed here in complete analogy to individual curves. In the context of our primary application—arrangement computation—this means, for example, that the expensive analysis of a tangential intersection of f and g is performed only if there actually are segments of both f and g containing the intersection's event x-coordinate in their x-ranges.

The `Event2_info` structure for a two-curve event is defined as follows:

```
struct Event2_info {
    //! for CURVE1 and CURVE2: event's arc number or -1
    int arc_at[2];
    //! arcs below event, as a sequence of CURVE1, CURVE2 values
    std::vector<Curve_index> below;
    //! arcs above event, as a sequence of CURVE1, CURVE2 values
    std::vector<Curve_index> above;
};
```

The source code for the analysis of two-curve events is contained in the separate file `CbX/compute_event2_info.h`.

6.2.3 NumeriX

Let us now take a quick look at the main algebraic operations underlying CubiX. They are implemented in the NumeriX library. We have described their functionality in Chapter 2.

`NiX/Polynomial.h` offers polynomials and algebraic operations on them, including the computation of gcds and resultants from subresultant polynomial remainder sequences. As an alternative to that, (sub)resultants can also be computed using `NiX/bezout_matrix.h` or `NiX/sylvester_matrix.h` as determinants of the respective matrices, drawing on `NiX/Linear_algebra.h` and friends.

We have chosen the latter alternative: Resultants of curves are computed by evaluating the determinant of their Bezout matrix (§2.3.1) using the division-free method of Berkowitz [R01]. Subresultants are computed similarly from the corresponding Sylvester submatrix (§2.3.3). An empirical justification is given in §6.3.5.

The recursive representation of a bivariate polynomial $f(x,y)$ as univariate polynomial in y over univariate polynomials in x over the integers (or whatever coefficient type) is problematic when operations w. r. t. the inner variable x are needed. The file `NiX/bivariate_polynomial_hacks.h` contains some aids for circumventing such problems without runtime penalties.

For example, the function `substitute_x()`to substitute a number a for the inner variable x is implemented as follows: Take the const iterator range over the coefficients of y. Use `LiS::Mapping_iterator` with `std::const_mem_fun1_ref_t` and the evaluation member function to turn that into an iterator range whose elements are the values of the coefficients with a substituted for x. Then construct $f(a,y)$ as the polynomial with that iterator range as coefficient sequence.

`NiX/Algebraic_number.h` implements the real algebraic numbers of §2.4.4, featuring the automatic simplification to a rational number or to a one-root number in a separation-bound number type (cf. §2.4.3).

Wherever possible, we have applied the idea of a *modular filter* mentioned in §2.2.4 and §2.4.4: A gcd computation of two univariate polynomials is not performed if their Bezout determinant, evaluated modulo the “random” prime number 67111067, is non-zero, because their gcd is then known to be constant.

This speeds up the factorization of resultants $R \in \mathbb{Z}[x]$ by multiplicities (which begins with computing $\gcd(R,R')$, see §2.2.3) and the comparison of algebraic numbers (which includes the gcd computation of their defining polynomials). Michael Hemmer has implemented the necessary modular arithmetic following [BE$^+$99] in `NumeriX/src/Modular.C` and related files.

The comparison of `Algebraic_number`s is sped up further by *pre-refining* their isolating intervals up to 16 times before the modular filter and maybe a gcd computation are invoked. (There is no strict justification for the choice of the number 16.)

6.2.4 SweepX

There are several points of interconnection between CubiX and SweepX. One major interconnection is the definition of algebraic points and segments using SoX/GAPS: The class templates `Algebraic_point_2` and `Algebraic_segment_2` implement the degree-independent ideas of Section 5.2 of how to define points, segments, and their predicates/constructions in terms of curves and curve pairs. We instantiate them with `Cubic_pair_2` as template argument, which also brings in the matching curve class `Cubic_curve_2`. The interface requirements imposed by the point and segment templates are laid down in the concepts AlgebraicCurve_2 and AlgebraicCurvePair (see reference documentation).

One part of the interface requirements is the provision of slices of a curve pair (cf. §4.2.9). Since the construction of a slice for an interval between events or for a two-curve event is degree-independent, a general `Slice` class template is offered in SoX/GAPS. `Cubic_pair_2` makes use of it. This is another dependency of CubiX on SweepX.

From an abstract point of view, slices follow almost trivially from the analysis of a curve pair over an interval or over an event *x*-coordinate. In writing code, however, the author has found them to be a quite useful abstraction. For example, the lexicographic comparison of two points `p` and `q` in a `Compare_xy` function object has the following straightforward implementation in which all case distinctions from the geometric analyses are abstracted away:

```
if (p.is_identical(q)) return LiS::EQUAL;
LiS::Three_valued c = p.x().compare(q.x());
if (c != LiS::EQUAL) {
    return c;
} else {
    Algebraic_curve_2 f = p.curve();
    Algebraic_curve_2 g = q.curve();
    if (f.is_identical(g)) {
        return sign(p.arcno() - q.arcno());
    } else {
        const typename Algebraic_curve_pair::Slice&
            slice = Algebraic_point_2::curve_pair(f,g).slice_at(p.x());
        return sign(slice.arcno_to_pos(f, p.arcno())
                  - slice.arcno_to_pos(g, q.arcno())
        );
    }
}
```

As exemplified by the call to `f.is_identical(g)` above, the predicates and constructions of SoX/GAPS require that whenever the supporting curves of two objects (i. e. points or segments) are equal, they are even *identical*, meaning that they are represented by the same piece of memory. This is not only helpful for fast (in)equality tests; more importantly it ensures that geometric analyses made in some curve object are not repeated elsewhere in an equal but non-identical curve object. For the same reason, it is desirable to make sure that for every unordered pair $\{f,g\}$ of curves there is only one curve pair object. To achieve this, the construction of curve and curve pair objects is indirected through `Curve_cache` and `Curve_pair_cache` objects. Using a `Curve_cache` is the responsibility of the library user who constructs points and segments. In contrast to this, curve pairs are constructed by predicates and constructions on the fly, so a `Curve_pair_cache` object for their internal use is maintained as a static member of `Algebraic_point_2`. All references to it are made through the `curve_pair()` static member function (as in the code above).

The main functionality of SweepX is arrangement computation, as implemented by the `sweep_curves()` function template, using the generic sweep-line algorithm of Section 5.1. The function is parametrized with a traits class, aggregating the point and segment classes to be used together with their predicates/constructions. (It is not a requirement to use specializations of the SoX/GAPS point and segment classes as we do. Arrangement computation for conics, as implemented by EXACUS's ConiX library [BE+02] uses the same sweep code but supplies hand-crafted points, segments, predicates, and constructions.)

The function `sweep_curves()` takes a set of segments, given as an iterator range, and produces a LEDA planar map representing the arrangement, with vertices and edges labelled accordingly. For our purposes, the necessary traits class, which must be a model of CurveSweepTraits_2, is supplied by CubiX, of course. This brings us back to ...

6.2.5 CubiX, part two

All the parts described above are put together in the `Cubic_curve_sweep_traits_2` class template. When instantiated with `LEDA_arithmetic_traits` (as defined in `NiX/Arithmetic_traits.h`), it yields a model of the CurveSweepTraits_2 concept, wrapping up CubiX ready for use with `SoX::sweep_curves()`.

But there are two more ingredients needed to compute the arrangement of a set of curves. The first is easy: In order to obtain sweepable segments from a curve, use the `curve_to_segs()` function. It implements the algorithm step **Curve to Segments** from §5.3.2.

The second is more involved: coordinate changes. C++ exceptions are used to signal the violation of position conditions imposed in §4.1.1 and §4.2.1. The caller of `SoX::sweep_curves()` must be prepared to catch all exceptions derived from CubiX's virtual exception base class `Ex_any`, see `CbX/exceptions.h`, and react accordingly. For subclasses of `Ex_reshear`, this means to perform a random shear of the coordinate system (see §4.3.2). Our shearing strategy is encapsulated in a `Random_shear` object; let us call it `shear`.

The internal state of `shear` is the currently chosen change of coordinates, and `shear` can be used as a function object: To transform a bivariate polynomial `f` into the new coordinate system, write `shear(f)`. (Of course, this must happen before a `Cubic_curve_2` object is constructed from the polynomial `f`.) A typical piece of code might look like this:

```
typedef NiX::LEDA_arithmetic_traits AT;
typedef CbX::Cubic_curve_sweep_traits_2<AT> CST;
typedef CST::Segment_2::Algebraic_curve_2 Curve_2;
CbX::Random_shear<AT::Integer> shear;

bool arrgmt_done = false;
do try {
    SoX::Curve_cache<Curve_2> cc;
    std::vector<CST::Segment_2> segs;
    for (int i = 0; i < input_poly.size(); i++)
```

```
        curve_to_segs<CST::Segment_2>(
                cc(shear(input_poly[i])), back_inserter(segs)
        );

    leda::GRAPH<CST::Point_2, CST::Segment_2> arrgmt;
    SoX::sweep_curves(segs.begin(), segs.end(), arrgmt, true, CST());
    arrgmt_done = true;

    // ... now use computed arrgmt ... //

} catch (const CbX::Ex_reshear& e) {
    shear.random(); // choose new random coordinates
} while (!arrgmt_done);
```

Shearing back the data coming out of the arrangement computation is also done with `shear`, using other suitably overloaded versions of the function-call operator.

To ease testing, the initial change of coordinates chosen by `Random_shear` is the identity map. Only if arrangement computation in the original coordinate system fails, a random shear is performed.

6.2.6 Demo programs

CubiX also contains executable demo programs in the `CubiX/demos` directory; notably the graphical `xcubi` demo written by Elmar Schömer, in which all parts of the current implementation are combined and can be tested. (The other demos only have a textual interface and are intended as low-level debugging aids. The `gen_cubics` and `perturb_cbx` programs are discussed in §6.3.2 and §6.3.3, resp.)

`xcubi`'s overall mode of operation is as follows: A set of cubic curves is maintained. Each curve is displayed using an approximate algorithm by Taubin [T94] based on `double` arithmetic which recursively subdivides the visible part of the plane down to individual pixels and colours each pixel that is known to contain a piece of the curve. The graphical display can be panned and zoomed. See [MS+02] for a survey of plotting methods.

A cubic curve $f \in \mathbb{Z}[x,y]$ is not input as a polynomial; instead, a set of interpolation points $\{(x_i,y_i)\}_i \subseteq \mathbb{Z}^2$ is given. (Using $\mathbb{Z}^2$ instead of $\mathbb{Q}^2$ is not a limitation, up to scaling by a common denominator.) Each point yields a linear condition $f(x_i,y_i) = 0$ on the ten unknown coefficients of a generic cubic curve f. In general position, exactly nine points are sufficient to uniquely and consistently determine a cubic curve (i. e. a polynomial up to a constant factor). Solving the resulting homogeneous linear system yields the desired curve.

Interpolation points can be specified graphically, or read from a `*.cbx` file. The `data` subdirectory contains many of them. See `xcubi.html` for a description of `xcubi`'s user interface: how to create, load, save, and sweep cubics.

Some of these linear conditions prescribing interpolation points can be exchanged for linear conditions prescribing differential information. To do so, assume that (x_1,y_1) is a point required to be on f, and assume further that it lies within an arc of f. Then there is an implicit function $y = \varphi(x)$ whose graph coincides with f around (x_1,y_1). Thus we obtain for all x around x_i that

$$0 = f(x,\varphi(x)). \tag{6.1}$$

Differentiating both sides with respect to x yields

$$0 = f_x(x,\varphi(x)) + f_y(x,\varphi(x)) \cdot \varphi'(x) \tag{6.2}$$

by the chain rule. We already know $(x_1,\varphi(x_1)) = (x_1,y_1)$. Picking a value for $\varphi'(x_1)$ turns (6.2) into a linear condition on the coefficients of f prescribing that slope. One can continue in this fashion to supply values for higher derivatives (curvature etc.). Differentiating (6.2) once more yields

$$\begin{aligned} 0 = \; & f_{xx}(x,\varphi(x)) + 2 f_{xy}(x,\varphi(x)) \cdot \varphi'(x) + f_{yy}(x,\varphi(x)) \cdot (\varphi'(x))^2 \\ & + f_y(x,\varphi(x)) \cdot \varphi''(x). \end{aligned} \tag{6.3}$$

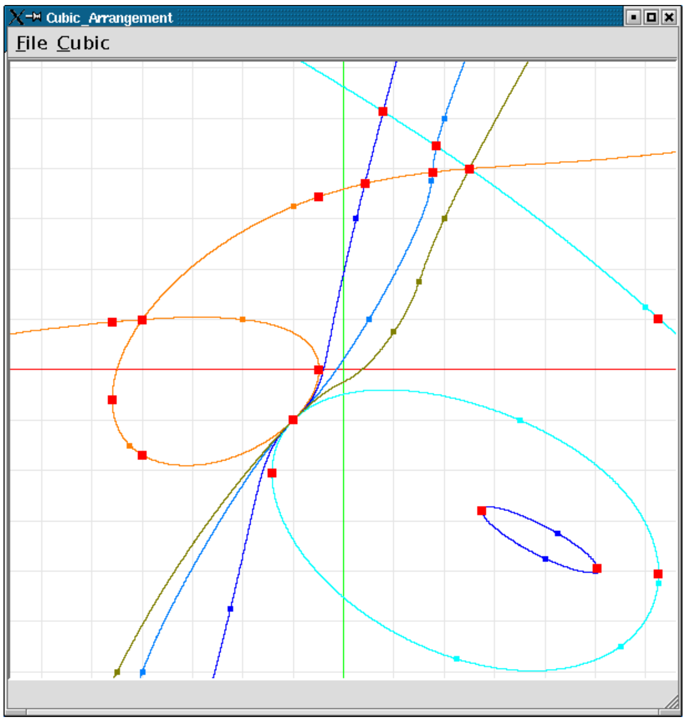
For `xcubi`, this has been done up to eighth derivatives, and the denominators of $\varphi^{(k)}(x_1)$ values have been cleared in the equations to keep their coefficients integral. In fact, this formulation allows us to use a slope of $\frac{1}{0} = \infty$ to prescribe a vertical tangent. That together with vanishing curvature $\varphi''(x_1) = \frac{0}{1}$ yields a vertical flex.

A careful analysis of the resulting linear equations shows that prescribing finite slope and infinite curvature yields two linear equations which are equivalent to the vanishing of f_x and f_y at (x_1,y_1). Heuristically, this is clear from the equations above: When $\varphi''(x_1) = \infty$, (6.3) implies $f_y(x_1,y_1) = 0$; and then $\varphi'(x_1) < \infty$ and (6.2) imply that $f_x(x_1,y_1) = 0$. Hence this way of specifying differential information can also be used to demand a singularity of multiplicity ≥ 2. In summary, it allows us to create a wide range of geometric degeneracies, including high multiplicities of intersections, curves with singularities, intersections involving singularities, many curves running through a common point, vertical flexes, and x-extreme points at forbidden places.

The nearby figure shows a screenshot of `xcubi` with a degenerate arrangement of five cubic curves which meet in one common point with intersection multiplicities 2 and 3. One of the curves is singular. This set of curves comes from the file `data/triple.cbx`. The small points are interpolation points. The bigger squares are the vertices of the computed arrangement. (On current hardware, this computation takes a second or less.) Besides intersection points, the vertices include one-curve events and points covertical to them on the same curve, which is a consequence of splitting input curves into sweepable segments.

The input of a cubic curve f with several components, one of which is to be a line g, is possible by interpolating through four points lying on g: By Bezout's

Theorem 3.2.4, g would not intersect f in these four points, unless g is a component of f. The rest $h = f/g$ of f will be a non-degenerate conic in general, unless there are three more collinear points that force h to be a line pair (by the same reasoning).



The `xcubi` demo showing the file `triple.cbx`.

The contents of `CubiX/demos/xcubi/`, in particular interpolation and visualization, are the work of Elmar Schömer. The author's part of the demo and the link to arrangement computation with CubiX and SweepX is contained in the header file `CbX/compute_arrangement.h`. That file's `compute_arrangement()` function prints various timings and statistics. As a consistency check, the planar map returned by `SoX::sweep_curves()` is verified to be a combinatorial planar map by testing that its genus $\frac{1}{2}(E - N - (F + I)) + C$ equals 0. (The capital letters denote the number of undirected e̲dges, n̲odes, f̲ace cycles, i̲solated nodes, and c̲onnected components in the arrangement.)

`xcubi` implements random shearing. You can observe this from the corresponding messages in the terminal window and from the vertices for x-extreme points, which are slightly shifted along the curve when an arrangement has been computed in a sheared coordinate system.

6.3 Benchmarks

6.3.1 General Remarks

To demonstrate the practical applicability of our approach, we have measured running times for arrangement computations of entire cubic curves in various instances. We used `xcubi` on a 1.2 GHz Pentium III Mobile CPU with 512 KB of cache under Linux. The executable was compiled with the GNU C++ compiler `g++` version 3.1 with optimization (`-O2`) and with all assertion statements switched off (`-DNDEBUG`). LEDA 4.4 was used for the exact number types and the data structures used by `sweep_curves()`.

The times have been measured with `LiS::User_timer`, which reports the running time allocated by the operating system, and they cover all steps of the algorithm beginning with the construction of `Cubic_curve_2` objects from polynomials and ending with the arrangement being returned by `sweep_curves()`. All examples can be computed in the original coordinate system, and no shearing has been performed.

Since `sweep_curves()` uses a randomized data structure internally to maintain X- and Y-structure (viz. `leda::skiplist`), there is a small variance in the measured running times. All running times mentioned below are rounded averages over five runs of the program (or three, for some instances with large times and small variance).

Two quantities have been used to measure the complexity of the input: the number n of curves; and the bit length of the largest coefficient in every curve, averaged over all n curves.

Clearly, the amount of work required from the algorithm depends on the complexity of the computed arrangement, i. e. the output. We measure this by the number of nodes and (directed) edges in the computed planar map. The cost of the algorithm is dominated by the curve pair analyses of Section 4.2, hence it makes sense to look at the number of analyses of intersection points, too, in particular for degenerate arrangments with nodes of high degree in the planar map.

6.3.2 Generating input data

The program `CubiX/demos/gen_cubics` generates random sets of n cubic curves in the `*.cbx` file format for `xcubi`; that is, it defines each curve by interpolation points and values of derivatives in some of those points, see §6.2.6. Various parameters of the generation process can be modified.

A `gen_cubics` run starts with the random choice of *interpolation points* on the integer grid $\{-128,\ldots,127\}^2$ and values of derivatives for each point. By default, there are $9n$ interpolation points chosen for n curves. The user can vary this default by specifying which percentage of $9n$ to use.

Then, n curves are generated from these interpolation points and their derivative values in the following fashion:

(1) With a user-defined probability p_{sing}, randomly choose a first interpolation point (x_1, y_1) and make it a singular point by prescribing finite slope and infinite curvature (see §6.2.6). This yields three conditions.

(2) Repeat the following steps until nine conditions altogether have been accumulated:

(2.1) Randomly choose an interpolation point (x_i, y_i). This yields one further condition.

(2.2) With a user-defined probability p_r prescribe that point's r-th derivative value for the curve, for $1 \leq r \leq 8$. Each derivative value yields one further condition.

By default, all user-defined probabilities are zero.

When measuring running times, we want to obtain average statements for several instances created with the same sets of parameters; on the other hand we want to refer to instance-specific data like the number of edges. Therefore, we have created five (or three) instances for each set of parameters and then used the one with median average running time.

The data files are included with the software; their file names indicate the series to which they belong (see below) and the arguments given to `gen_cubics` for their generation.

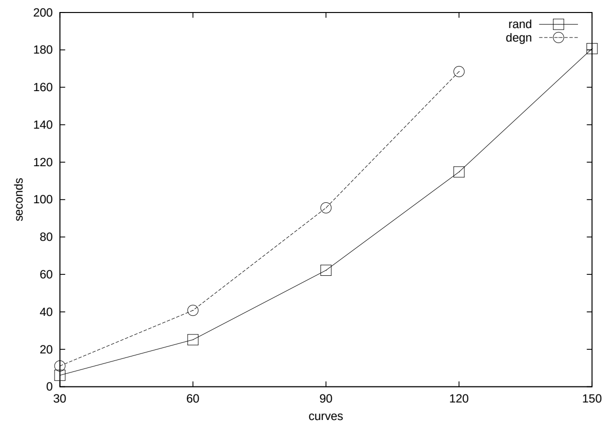
6.3.3 Growth of input data sets

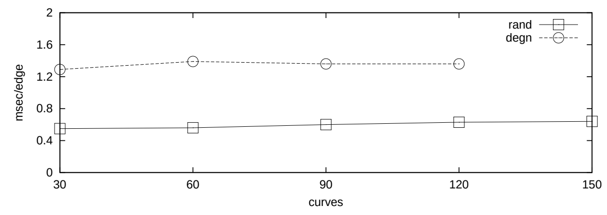
The first set of measurements looks at the growth of the running time for arrangement computation as a function of the number n of input curves. We have tested two series of instances for growing n:

- The `rand` instances consist of n random curves interpolated through $9n$ random points. There are no intersections of multiplicity larger than 1 and no singular points. We have considered instances for $n = 30, 60, 90, 120, 150$.
- The `degn` instances consist of n degenerate curves interpolated through 54 points. Each curve has a $p_{\text{sing}} = 0.20$ chance of being singular. In each interpolation point, slope is prescribed with certainty ($p_1 = 1.00$), whereas curvature is prescribed with probability $p_2 = 2^{-1/2} \approx 0.71$, so that any two

curves through the point (and not having a singularity there) have intersection multiplicity 2 or 3 with probability $\frac{1}{2}$ each. Furthermore, the small fixed number of interpolation points causes nodes of high degree. There are instances for $n = 30, 60, 90, 120$.

It was necessary to fix the number of interpolation points and not the ratio of interpolation points to n for the `degn` examples to keep the probability constant that any two curves have a degenerate intersection point. This makes sure that the density (relative frequency) of degenerate intersections among all intersections remains constant.





The adjacent figure shows the graphs of average running times as functions of the number n of input curves. The growth is roughly quadratic, as is the size of the computed arrangements, and gives an indication of the running time as a function of the input size.

The next figure shows the average running time divided by the number of edges. This gives an idea of the running time as a function of the output size, and it seems to match the linearithmic output-sensitive complexity of the straight-line segment sweep [MN99, 10.7.2].

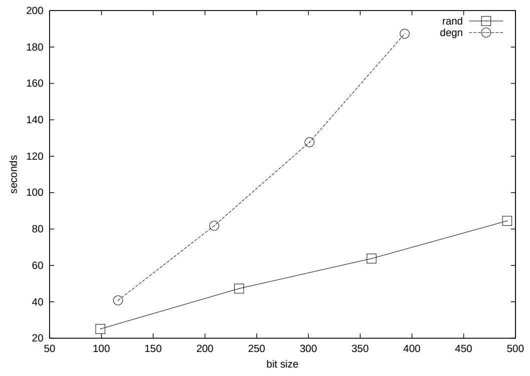
The tables below give the raw numbers from the measurements. Segs is the number of sweepable segments. Time is given in seconds. Bits is the average bit length of a curve's longest coefficient.

series	n	segs	nodes	edges	bits	time
`rand`	30	226	2933	11083	99	6.1
`rand`	60	454	11417	44440	99	25.1
`rand`	90	680	26579	104474	100	62.2
`rand`	120	940	46117	181978	100	114.8
`rand`	150	1226	71594	283144	99	180.7

series	n	segs	nodes	edges	bits	time
`degn`	30	243	2313	8604	116	11.1
`degn`	60	534	7627	29284	116	40.8
`degn`	90	722	17983	70378	120	95.6
`degn`	120	1027	31504	123814	114	168.4

Almost all of the measured running time is spend in `sweep_curves()`. The preceding phase of splitting of curves into sweepable segments, as part of which all one-curve analyses take place, is well below a second even for the largest instances.

A second set of measurements addresses growth in the coefficients' bit lengths. We have taken the `rand_60` and `degn_60` instances from above and modified them with `CubiX/demos/perturb_cbx` to increase bitlengths while keeping the combinatorial structure (almost) unchanged. Since interpolation data for `xcubi` has to be integral, we first scaled each interpolation point (x_i, y_i) by a factor $s = 100$, $10\,000$, or $1\,000\,000$, and then perturbed it to $(sx_i + \varepsilon_x,\ sy_i + \varepsilon_y)$ with ε_x, ε_y chosen randomly from $\{-10, \ldots, 10\}$. (All occurrences of an interpolation point were mapped in the same way such as to preserve degeneracies.) We have not perturbed derivative values, but to transform them in accordance with the scaling transformation, each value of an r-th derivative $\varphi^{(r)}(x_i)$ has been scaled by division by s^{r-1}.



Graphs of average running times as a function of average maximal coefficient bit length (left), and tables of running times (below).

`rand_60`	segs	nodes	edges	bits	time
original	454	11417	44440	99	25.1
with $s = 10^2$	454	11437	44520	233	47.3
with $s = 10^4$	454	11417	44440	361	63.8
with $s = 10^6$	454	11417	44440	492	84.5

`degn_60`	segs	nodes	edges	bits	time
original	534	7627	29284	116	40.8
with $s = 10^2$	534	7639	29332	209	81.8
with $s = 10^4$	534	7627	29284	301	127.7
with $s = 10^6$	534	7627	29284	393	187.3

6.3.4 Costs of various degeneracies

The `rand` series of input data sets realizes the case of generic input curves without any degeneracies. The `degn` series contains some mix of degeneracies. Below we give an indication of the individual cost of the following three kinds of degeneracies. The corresponding data files form the `idegn` series.

- Twofold: 60 curves are interpolated through 81 points, and p_1 is set to 1.00 to create intersections of multiplicity 2. Hence each curve needs 5 interpolation points.
- Threefold: 60 curves are interpolated through 48 points, and we set $p_1 = p_2 = 1.00$ to create intersections of multiplicity 3, so that each curve needs 3 interpolation points.
- Singular: 60 curves are interpolated through 21 points. Each curve has a chance of $p_{\text{sing}} = 0.50$ to be singular, so the expected number of interpolation points used per curve is 8.

The number of interpolation points may seem arbitrary, but these numbers were chosen on purpose: The expected number of interpolation points used for a curve in the `degn_60` instance is approximately 3.3 out of 54, so there are about 16 times more interpolation points in total than used by one curve. This fraction was stabilized for the twofold and threefold intersection instances to achieve comparable densities of degenerate intersection points in the resulting arrangements.

Singularities, however, can appear at most once on each curve, so intersections involving them are not so frequent in `degn`. Therefore, the number of interpolation points was chosen to be much smaller for the singular instance in order to attain a reasonable density of degenerate points and a measurable speed difference.

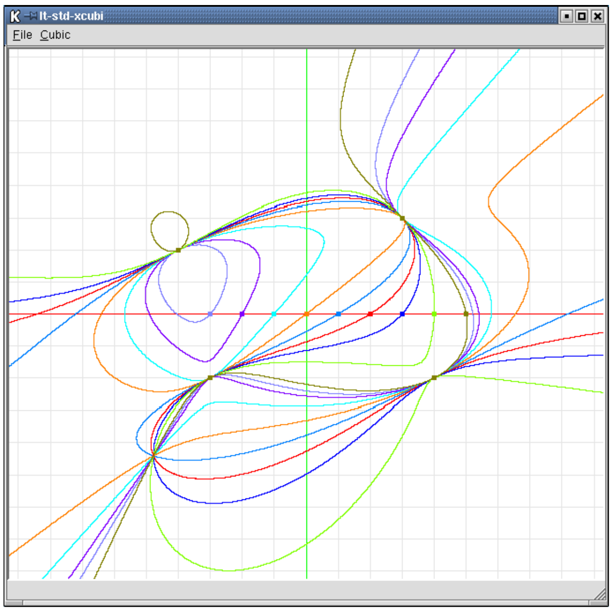
The following table gives input/output sizes, running times, and the numbers of intersection point analyses (which may vary slightly due to the randomized X- and Y-structures), both total and distinguished by kinds of degeneracies (r-s = intersection point is regular on one curve and singular on other; s-s = intersection point is singular on both).

instance	segs	nodes	edges	time	total i.	2fold	3fold	r-s	s-s
`rand_60`	454	11417	44440	25.1	11000	0	0	0	0
`degn_60`	534	7627	29284	40.8	7308	154	118	30	0
Twofold	516	9167	35500	41.0	8946	291	0	0	0
Threefold	478	7308	28046	29.3	6937	0	253	0	0
Singular	499	7532	29752	34.0	11045	0	0	626	38

A few remarks on the interpretation: The additional cost incurred by these relatively small numbers of degenerate intersections cannot be attributed solely to the analyses of the degenerate intersection points themselves. Before the analysis of any intersection point of f and g takes place, $R_{fg} = \operatorname{res}(f, g, y)$ is factored by multiplicities, and this is expensive, since R_{fg} is not square free in the presence of degenerate intersections, i. e. there is no shortcut like the modular filter in the square-free case. This presumably dominates the additional cost of threefold intersections.

Intersections of multiplicity 2 are a particularly expensive kind of degeneracy, since their treatment involves a Jacobi curve and three additional resultants, including isolation against their zeroes and potentially prior factorizations by multiplicities (see §4.2.5). We use them because an intersection point of multiplicity 2 may be irrational and may have an algebraic degree up to 4. In that case, however, the computations with the additional resultants can be amortized over up to 4 points. In the example above, however, most curve pairs have only one twofold intersection, which is then rational by Proposition 2.4.3, so that proceeding with rational arithmetic where possible would be faster for this input data.

A more appropriate setting for Jacobi curves is the adjacent manually constructed example `4x2` of nine cubic curves intersecting in five points, four of which are intersection points of multiplicity 2. (The fifth intersection point is not an interpolation point but occurs automatically — what a nice instance of the Theorem of the Nine Associated Points! [G98, 17.1]) For comparison, there is also a perturbed version `4x2_pert` with 146 simple intersections of exactly two curves each. Here the additional cost of using Jacobi curves is less bad.



The following table contains the average running times in seconds. About half of the time difference is accounted for by the factorization of resultants by multiplicities.

instance	segs	nodes	edges	bits	time	total i.	2fold
`4x2`	65	79	220	35	0.80	69	56
`4x2_pert`	61	216	706	48	0.33	146	0

6.3.5 Effect of optimizations

We have described certain implementation choices in §6.2.3. We now provide some empirical evidence to justify them, at least for the case of input data free of degeneracies where the modular filter can show its full strength. The following variants of the implementation have been compared:

- The *standard* variant as described in §6.2.3.
- The *PRS-res* variant in which resultants of curves are computed from polynomial remainder sequences using the subresultant algorithm [C93, 3.3] and not as determinants.

- A *modular* variant with modular filter but without pre-refinement of isolating intervals.
- A *basic* variant without modular filter and without pre-refinement.

The following tables contain average running times in seconds and convey a clear message.

instance	*standard*	*PRS-res*
`rand_30`	6.1	7.1
`rand_150`	180.7	204.3

instance	*standard*	*modular*	*basic*
`rand_30`	6.1	7.5	65.2

Note that the modular filter helps to avoid gcd computations both in the comparison of algebraic numbers and in the factorization of resultants by multiplicities (including those for Jacobi curves), whereas pre-refinement addresses only the first issue. Also note that algebraic numbers are not only used for event x-coordinates, but also for sorting arcs over intervals between two-curve events where the algebraic numbers are never equal.

The executables for the variants of the algorithm were created by adjusting the following preprocessor macros via compiler `-D` switches:

- To switch off pre-refinement, remove `-DNiX_REFINEMENTS_BEFORE_GCD=16` from the automatically generated `Makefile`.
- The modular filter is deactivated by giving `-DNiX_MODULAR_FILTER_OFF`.
- The function called in CubiX for resultant computation can be changed by setting `-DCbX_RESULTANT=`... accordingly.

A textual output of the `leda::GRAPH` computed by `sweep_curves()` is printed by `xcubi` when compiled with `-DCbX_DUMP_LEDA_GRAPH`. The numbers of intersection point analyses in the tables above were determined using `-DCbX_PRINT_INTERS` which makes each intersection point analysis print a message.

6.3.6 A little bit of profiling

We have compiled the standard variant of `xcubi` on a SunBlade 100 (clocked at 500 MHz) for profiling (`-pg -g -O2 -DNDEBUG`) with a special version of LEDA 4.4.1 (compiled with C++ code in place of the optimized integer arithmetic so that constructing the call graph works), and we have run it on `rand_30`.

There is one basic operation which clearly dominates the running time: To sweep these 30 curves, 4 857 525 multiplications of `leda::integers` were performed, accounting for 48 % of the overall running time of `compute_arrangement()` (excluding profiling overhead). Clearly, the cost of a multiplication depends heavily on the bitlengths of the factors. However, `gcc`/`gprof` distribute the time spent in a function evenly among all calls, so all subsequent fractions of running time are to be taken with a fair amount of scepticism.

`compute_arrangement()` spends 94 % of its time in `sweep_curves()`, mainly for finding and handling intersections. Among the member functions of a curve pair object, `arcs_over_interval()` seems the most expensive: 54 % of the running time is supposedly expended for it. (Since all intersections are simple, this covers most of the effort involved in finding intersections based on comparing sequences of arcs.) A quarter of these 54 % is needed for constructing the univariate polynomials $f(r_i,y), g(r_i,y)$, half is spent with Uspensky's algorithm on them (accounting for 94 % of all invocations of Uspensky's algorithm), and the remaining quarter on expressing and comparing the roots as `Algebraic_number`s.

All comparisons of `Algebraic_number`s together account for 35 % of the running time. Almost no gcd computations had to be performed, thanks to the modular filter.

Chapter 7

Conclusions

Results and Future Work

We have presented a method to compute the planar arrangement of segments of algebraic curves of degree up to 3 which is ...

- Exact and complete: The presented method can compute the mathematically correct result in all cases, including all degeneracies.
- Practically feasible: We need several seconds for non-degenerate arrangements with thousands of vertices. That is, roughly speaking, the same order of magnitude reported four years ago in [MN99, 10.7.4] for straight-line arrangements. (Of course, this is also a result of faster hardware, but available hardware defines what is practically feasible and what not.)
- Novel. To our knowledge, we are the first to present a method combining both of the two items above.
- Heading towards Availability. The implementation has been written with the goal to produce reliable and maintainable code that can form the basis of a publicly available EXACUS component at some point in the future.

But the author does not expect this to be the end of the story. It is a first success that leaves much room for future work, and completing the software is the least.

Firstly, we do not take any advantage yet of floating-point methods once symbolic calculation has indicated their applicability. In particular, ordering the arcs of a curve pair over an interval between events ought to be a problem which is numerically extremely well-conditioned most of the time, yet we use unconditionally the heavyweight algebraic numbers of §2.4.4 with exact rational interval boundaries and the cumbersome refinement by bisection. Similar remarks apply to the comparison of x-coordinates once the modular filter has determined their inequality.

Secondly, the idea of combining guaranteed floating-point approximations with separation bounds to yield exact arithmetic, as exemplified by the `leda::real` and `CORE::expr` number types (see §2.4.3), has only been applied in a very limited form so far, namely as much as it is available ready for use with square root expressions in the form of these number types, and its applicability to the problem at hand has not yet been examined in full generality.

In both respects, [BFMSS01]-strength `leda::real`s with the diamond operator are worth considering: Concerning the second paragraph, the diamond operator allows to compute exactly in the field of all real algebraic numbers and nicely encapsulates one rule set for separation bounds. These separation bounds can get very large, however, so it is not clear how much we can hope for in this direction. Nevertheless, a limited application might be restricted to the cases of the first paragraph: Use the diamond operator and the comparison of diamond expressions as an elegant way to reuse the underlying numerical methods for the comparison of distinct numbers.

On a broader scale, the question arises how to extend our ideas to plane algebraic curves of arbitrary degrees. In the absence of singularities, Nicola Wolpert has addressed this in [W03]. For singularities, the method described in this text relies very much on the specific properties of cubic curves and their textbook classification. It does not seem promising to proceed further on a degree-by-degree basis, except for special contexts (such as the restricted class of quartic curves arising from projected intersections of spatial quadrics [W02]), because the more diverse the singularities can become, the more important is it to treat them by a general method and not by case distinctions.

Speaking on the most general level, it is not at all clear what practically applicable approaches to the fundamental problems of exact and efficient computational geometry for curved objects will turn out to be best in the course of future research and experience. It is inconceivable that our considerations should already have exposed them; but the author hopes that our results can serve as a stepping stone towards them.

Colophon

The author gratefully acknowledges the following sources of figures:

- The plots of the singular irreducible cubic curves appearing in Chapter 3 have been produced using the fine program `surf` [Surf].
- Many figures in Chapter 4 are modified versions of figures produced by Nicola Wolpert for [ES$^+$02] with `xfig`.
- The other figures are screenshots or have been created by the author using `xfig` and `gnuplot`.

The text has been typeset with $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX, using `pslatex`, `epsfig` and `wrapfig`.

Appendix A

Mathematical Addenda

A.1 Polynomials over Non-UFDs

We have repeatedly used Gauss' Lemma (Proposition 2.2.7) and its consequences to compute with polynomials over the integers instead of the rationals, which is less expensive in terms of running time. In particular, Corollary 2.2.9 told us the following: When we look for rational factors g of some non-zero polynomial $f \in \mathbb{Z}[x]$, we can restrict ourselves to $\mathbb{Z}[x]$, because the additional factors to be found in $\mathbb{Q}[x]$ are just constant multiples.

We will give two examples to demonstrate that this is not a general property of quotient fields and to exemplify two ways in which this can go wrong for rings $\mathbb{Z}[\sqrt{d}]$. These rings occur naturally when trying to compute in quadratic number fields $\mathbb{Q}(\sqrt{d})$ without fractions.

As the first example, consider $R = \mathbb{Z}[\sqrt{5}]$. We have $\mathrm{Q}(R) = \mathbb{Q}(\sqrt{5})$. The polynomial $f = x^2 - \sqrt{5}x + 1$ factorizes over $\mathrm{Q}(R)$ as $f = (x - \frac{1}{2}(\sqrt{5}-1))(x - \frac{1}{2}(\sqrt{5}+1))$. However, it is irreducible over R: Assume $f = gh$ with $\deg(g) = \deg(h) = 1$. We have $1 = \ell(f) = \ell(g)\ell(h)$ such that the leading coefficients are units, and hence we can assume w. l. o. g. that $g(x) = x - r$ and $h(x) = x - s$ for some $r, s \in R$. It follows that r and s would have to be the roots $\pm\frac{1}{2} + \frac{1}{2}\sqrt{5}$ of f which clearly are not elements of R.

To see what goes wrong, observe that $4f = 4x^2 - 4\sqrt{5}x + 4 = (2x - \sqrt{5} + 1)(2x - \sqrt{5} - 1)$ is divisible by $4 = 2^2 = (\sqrt{5}+1)(\sqrt{5}-1)$, but its factorization into two linear polynomials splits this constant factor in two incompatible ways, which is only possible because $\mathbb{Z}[\sqrt{5}]$ is not a UFD.

A mathematician would probably remark that this is a well-known problem whenever $d \equiv 1 \pmod 4$ [H23, §29], and the fundamental error was to use $\mathbb{Z}[\sqrt{d}]$ in the first place instead of the ring of algebraic integers of $\mathbb{Q}(\sqrt{d})$ (i. e. the subring of $\mathbb{Q}(\sqrt{d})$ containing all zeroes of monic polynomials), which is $\mathbb{Z}[\frac{1}{2}(1+\sqrt{d})]$. Indeed, this eliminates the problem for monic linear factors.

So let us consider a second example which does not fail in this simple fashion: $d = 10$. We refer to [HW79, 14.6] [H64, 16.6] for the fact that 2, 5, and $\sqrt{10}$ are irreducible elements of $\mathbb{Z}[\sqrt{10}]$. It follows immediately that $\mathbb{Z}[\sqrt{10}]$ is not a UFD, because $10 = \sqrt{10}\cdot\sqrt{10} = 2\cdot 5$ factors in two essentially different ways. Consider the polynomial $f = 10x^2 + 7\sqrt{10}x + 10 = (2x+\sqrt{10})(5x+\sqrt{10})$. Due to the irreducibility of 2, 5, and $\sqrt{10}$, the two linear factors on the right-hand side have coprime coefficients, whereas the coefficients of their product have a common factor $\sqrt{10}$. As a consequence, $f/\sqrt{10}$ is irreducible in $\mathbb{Z}[\sqrt{10}][x]$, but it has the two obvious linear factors in $\mathbb{Q}(\sqrt{10})[x]$.

It is still possible to compute gcds and resultants that conceptually refer to $\mathrm{Q}(R)[x]$ and actually just use polynomials from $R[x]$. But a greatest common divisor for $f, g \in R[x]$ regarded as elements of $\mathrm{Q}(R)[x]$ may no longer divide f and g within $R[x]$.

We conclude by remarking that Ernst Eduard Kummer (1810–1893) introduced the notion of an *ideal* to regain a form of unique factorization in non-UFDs [H23, §23], which allowed him to extended Gauss' Lemma by defining the content of a polynomial as the ideal generated by its coefficients [H23, §28]. This contains our definition of content as a special case, because in a principal ideal domain like $\mathbb{Z}$, the ideal generated by some elements is generated just as well by their gcd.

A.2 Degrees of Singularities

Proposition 2.4.3 allows us to strengthen the statement of Proposition 3.3.2 on the degree of singular points and to extend it to arbitrary algebraic curves.

Proposition A.2.1: *Let $f \in \mathbb{Q}[x,y]$ be an algebraic curve which has exactly n distinct singularities of multiplicity m in the complex affine plane $\mathbb{C}^2$, and let (r,s) be one of them. Then the algebraic degree of (r,s) is at most n.*

Reacll that the algebraic degree of (r,s) is the degree of the number field $\mathbb{Q}(r,s)$.

Proof: Multiplicity m (as opposed to $\geq m$) means that all partial derivatives of f of order up to $m-1$ vanish at (r,s) but some m-th partial derivative, say g, does not. Consider the following system $S \subseteq \mathbb{Q}[x,y,z]$ of polynomial equations:

$$\begin{aligned} S \;&:=\; \{h \mid h \text{ is an } i\text{-th partial derivative of } f,\ 0 \leq i < m\} \\ &\;\cup\; \{1 - zg(x,y)\} \end{aligned}$$

One solution is (r,s,t) where $t := g(r,s)^{-1} \in \mathbb{Q}(r,s)$. By Proposition 2.4.3 there are at least $d := \deg((r,s))$ conjugate solutions $(r',s',t') \in \mathbb{C}^3$. Since t' is uniquely determined by r,s as $t' = g(r',s')^{-1}$, this means that there are at least d different partial solutions (r',s'), and each of them is a singularity of f with multiplicity m. $\square$

Similar "frequency bounds degree" statements can be obtained for all other kinds of points that can be defined as solutions of polynomial (in)equalities over $\mathbb{Q}$.

The technique used above to encode an inequality constraint by means of an auxiliary variable is known as the *trick of Rabinovitch*.

A.3 Some rational conics without rational points

Proposition A.3.1: *Let* $F = x^2 + y^2 - rz^2$, $r \in \mathbb{Z}$ *with* $r \equiv 3 \pmod 4$. *The projective algebraic curve* F *does not contain any rational points.*

Proof: Any point $(a:b:c) \in \mathbb{P}^2(\mathbb{Q})$ can be written with integer coordinates $0 \neq (a,b,c) \in \mathbb{Z}^3$ such that $\gcd(a,b,c) = 1$. Assume $F(a:b:c) = 0$. This implies

$$a^2 + b^2 + c^2 \equiv 0 \pmod 4.$$

Modulo 4, we have $\overline{0}^2 = \overline{2}^2 = \overline{0}$ and $\overline{1}^2 = \overline{3}^2 = \overline{1}$. Therefore $a^2 \equiv b^2 \equiv c^2 \equiv 0$, so that 2 divides a, b, and c, contradicting their triple-wise coprimality. Hence no rational point exists on F. $\square$

This proposition is an immediate corollary to the Two Squares Theorem for arbitrary integers [HW79, Thm. 366].

Since F does not contain any rational points, neither does $G = F \circ M$ for any projective change of coordinates $M \in \mathrm{GL}_3(\mathbb{Q})$. Dehomogenizing G to $g \in \mathbb{Q}[x,y]$ yields an irreducible conic. For $r > 0$, it is an ellipse or hyperbola containing infinitely many points of $\mathbb{R}^2$ but none of $\mathbb{Q}^2$. The free choice of r and M provides an infinite supply of conics without rational points.

A parabola $g \in \mathbb{Q}[x,y]$ cannot occur in this way, since then $G \in \mathbb{Q}[x,y,z]$ would have the line at infinity, whose equation is $z = 0$, as a tangent [G98], giving rise to a unique and hence rational singularity of the cubic curve $zG(x,y,z) = 0$. (To apply Proposition A.2.1, dehomogenize w. r. t. x or y.)

A.4 Rational triangles with irrational vertices

Let us construct triangles $f \in \mathbb{Q}[x,y]$ with rational coefficients. It is straightforward how to do this for rational vertices: Just interpolate lines through any pair of them and multiply the three resulting sides. Clearly, all triangles with rational vertices can be constructed in this way. It is not so obvious how to construct all instances, or even just one instance, of a triangle with irrational vertices. That is the object of this section.

Let $f \in \mathbb{Q}[x,y]$ be a y-regular cubic curve which consists of three distinct line components $f = g_1g_2g_3$ that form a triangle with vertices s_1, s_2, and s_3 as in §3.3.2. Let $\{i, j,k\} = \{1,2,3\}$ be a triple of three distinct indices. The coordinates of s_i are $s_i = (\vartheta_i,\eta_i)$. The line g_k is uniquely determined by the two vertices s_i and s_j on it. The y-regularity of f extends to its component g_i by Proposition 2.2.1 and implies that $\vartheta_i \neq \vartheta_j$.

There is one polynomial $\eta(x) = e_2x^2 + e_1x + e_0 \in \mathbb{Q}[x]$, $e_2 \neq 0$ such that $\eta(\vartheta_i) = \eta_i$ for all $i = 1,2,3$. We have constructed it in §4.1.7 under the name $-\eta_2(x)$. This allows us to focus on the x-coordinates $\vartheta_1,\vartheta_2,\vartheta_3$ which we regard as indeterminates from now on. We have

$$\begin{aligned} g_k &= y + \frac{\eta(\vartheta_j) - \eta(\vartheta_i)}{\vartheta_i - \vartheta_j}x + \frac{\eta(\vartheta_i)\vartheta_j - \eta(\vartheta_j)\vartheta_i}{\vartheta_i - \vartheta_j} \\ &= y - (e_2(\vartheta_i + \vartheta_j) + e_1)x + e_2\vartheta_i\vartheta_j - e_0 \end{aligned} \tag{A.1}$$

Observe that g_k is invariant under exchanging $\vartheta_i \leftrightarrow \vartheta_j$.

Consider $f = g_1g_2g_3 \in \mathbb{Q}[\vartheta_1,\vartheta_2,\vartheta_3][x,y]$ and its monomials $a_{uv}x^uy^v$ with their coefficients $a_{uv} \in \mathbb{Q}[\vartheta_1,\vartheta_2,\vartheta_3]$. Any permutation $\sigma : \{1,2,3\} \to \{1,2,3\}$, $i \mapsto \sigma(i)$ acts homomorphically on polynomials involving the variables $\{\vartheta_i \mid i \in \{1,2,3\}\}$ by substituting $\vartheta_{\sigma(i)}$ for ϑ_i. But the polynomials $a_{uv} \in \mathbb{Q}[\vartheta_1,\vartheta_2,\vartheta_3]$ are invariant under this operation, because

$$\sigma(f) = \sigma(g_1)\sigma(g_2)\sigma(g_3) = g_{\sigma(1)}g_{\sigma(2)}g_{\sigma(3)} = f.$$

In other words: the a_{uv} are *symmetric* polynomials in $\vartheta_1,\vartheta_2,\vartheta_3$. This sets the stage for the following theorem [B96, 4.4] [L84, V.9]:

Theorem A.4.1: *(Fundamental Theorem of Symmetric Polynomials)*
Let K be a field. Let $f \in K[t_1,\dots,t_n]$ be a symmetric polynomial. Define the elementary symmetric polynomials $S_0,\dots,S_n$ in $t_1,\dots,t_n$ by

$$\sum_{j=0}^{n}(-1)^jS_jz^{n-j} = \prod_{i=1}^{n}(z - t_i) \tag{A.2}$$

where z is an auxiliary variable. (Observe $S_0 = 1$.) Then there exists a polynomial $\hat{f} \in K[x_1,\dots,x_n]$ such that $f = \hat{f}(S_1,\dots,S_n)$.

For our purposes, this means that the triangle $f = g_1g_2g_3$ can be rewritten as a polynomial in the coefficients of $h = (x-\vartheta_1)(x-\vartheta_2)(x-\vartheta_3)$. For values of $\vartheta_1,\vartheta_2,\vartheta_3$ from any triangle f, this polynomial h has to be rational, because $r := \operatorname{res}(f, f_y,y)$ has degree at most 6 by Bezout's Theorem and has each ϑ_i as a multiple root by §3.2.5, so that $r = \ell(r)h^2 \in \mathbb{Q}[x]$, which we can factor by multiplicitites (§2.2.3) to obtain $h \in \mathbb{Q}[x]$.

Therefore we can construct any y-regular rational triangle f by choosing a monic square-free polynomial $h \in \mathbb{Q}[x]$ that vanishes at the x-coordinates of the vertices, choosing a polynomial $\eta(x) = e_2x^2 + e_1x + e_0 \in \mathbb{Q}[x]$ to express their y-coordinates, and rewriting all coefficients $a_{uv}(\vartheta_1, \vartheta_2, \vartheta_3)$ of f as $\hat{a}_{uv}(h_2, h_1, h_0)$ in the coefficients of $h(x) = x^3 + h_2x^2 + h_1x + h_0$.

To obtain a triangle, the three vertices must not be collinear, meaning that the following determinant (the orientation predicate, cf. [MN99, 9.2.1]) must not vanish:

$$\begin{vmatrix} 1 & \vartheta_1 & \eta_1 \\ 1 & \vartheta_2 & \eta_2 \\ 1 & \vartheta_3 & \eta_3 \end{vmatrix} = \begin{vmatrix} 1 & \vartheta_1 & e_2\vartheta_1^2 + e_1\vartheta_1 + e_0 \\ 1 & \vartheta_2 & e_2\vartheta_2^2 + e_1\vartheta_2 + e_0 \\ 1 & \vartheta_3 & e_2\vartheta_3^2 + e_1\vartheta_3 + e_0 \end{vmatrix} = e_2 \begin{vmatrix} 1 & \vartheta_1 & \vartheta_1^2 \\ 1 & \vartheta_2 & \vartheta_2^2 \\ 1 & \vartheta_3 & \vartheta_3^2 \end{vmatrix}$$

The rightmost determinant is a Vandermonde determinant [L84, p. 298] and hence non-zero. So the non-collinearity requirement is equivalent to $\deg(\eta) = 2$.

With a modern computer algebra system at hand, the rewriting step for the coefficients of f can be performed using Gröbner bases:

1. Choose $\eta \in \mathbb{Q}[x]$ with $\deg(\eta) = 2$.

2. Let $f = g_1g_2g_3 \in \mathbb{Q}[x,y][\vartheta_1, \vartheta_2, \vartheta_3]$ with g_1, g_2, g_3 as in (A.1).

3. Take the variables $\vartheta_1 > \vartheta_2 > \vartheta_3 > h_2 > h_1 > h_0$ and order monomials lexicographically.

4. Take the elementary symmetric polynomials $S_1, S_2, S_3 \in \mathbb{Q}[\vartheta_1, \vartheta_2, \vartheta_3]$ according to their definition (A.2).

5. Compute a Gröbner basis G for the ideal

   $$I = (h_{3-j} - (-1)^j S_j(\vartheta_1, \vartheta_2, \vartheta_3) \mid j = 1,2,3)$$

   which reflects the definition of the h_i in terms of $\vartheta_1, \vartheta_2, \vartheta_3$.

6. Use G to compute the normal form of f modulo I. This eliminates the ϑ_i in favour of the h_i. (Maple automatically does this coefficient-wise. If your system does not, compute the normal forms of the coefficients a_{uv} individually.)

7. Choose the monic square-free polynomial $h \in \mathbb{Q}[x]$, $\deg(h) = 3$. For f, this means substituting rational numbers for the variables h_i occurring in the a_{uv}.

The result is a triangle $f \in \mathbb{Q}[x,y]$ with $\mathrm{res}(f, f_y, y) = h^2$ (up to constants), as desired, and the derivation of the method shows that all y-regular rational triangles can be obtained in this way, including those with irrational vertices: Just choose h accordingly. The assumed y-regularity is no restriction for the interesting cases where all vertices are irrational: If f is not y-regular, one component g_i is a vertical line whose equation can be obtained as $g_i = \mathrm{cont}(f) \in \mathbb{Q}[x]$ by taking a hierarchical view on $f \in \mathbb{Q}[x][y]$, and then s_i is rational (cf. §3.3.3).

The author was motivated to find a method like this to construct a particularly hard class of input curves for our algorithm: Triangles with two complex sides, having an irrational acnode of algebraic degree 3. For example, let $\eta(x) = x^2$ and $h(x) = x^3 + x - 1$. All rational zeroes of h must be integer by Proposition 2.4.5. But they must also divide the constant term 1, yet $h(\pm 1) \neq 0$, hence there are none, so that h is irreducible. Two of its zeroes are complex, because $h'(x) = 3x^2 + 1 > 0$ for all $x \in \mathbb{R}$. One obtains the rational triangle

$$f(x,y) = y^3 + y^2 + (x^2 - 3x)y + x^3 + x^2 - 2x + 1 \tag{A.3}$$

with an irrational acnode at $(0.6823278\ldots, 0.46557123\ldots)$.

Bibliography

[AS00] P. K. Agarwal, M. Sharir: “Arrangements and their applications”. In J.-R. Sack and J. Urrutia (eds.), *Handbook of Computational Geometry*, pp. 49–119, Elsevier, Amsterdam, 2000.

[A89] A. G. Akritas: *Elements of Computer Algebra*, Wiley, New York, 1989.

[A98] Mathew H. Austern: *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*, Addison-Wesley, 1998.

[BO79] J. L. Bentley, T. Ottmann: “Algorithms for reporting and counting geometric intersections”. *IEEE Transactions on Comp.* **C-28** (1979), pp. 643–647.

[BE$^+$02] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, E. Schömer: “A Computational Basis for Conic Arcs and Boolean Operations on Conic Polygons”. *Proc. European Symp. on Algorithms 2002* (ESA 2002), pp. 174–186, Springer LNCS 2461, Berlin, 2002.

[B98] Robert Bix: *Conics and Cubics: A Concrete Introduction to Algebraic Curves*, Springer UTM, New York, 1998.

[B96] S. Bosch: *Algebra*, 2nd ed., Springer, Berlin, 1996 (in German).

[BK86] E. Brieskorn, H. Knörrer: *Plane Algebraic Curves*, Birkhäuser, Basel, 1986. German original: *Ebene algebraische Kurven*, Birkhäuser, Basel, 1981.

[BE$^+$99] H. Brönnimann, I. Emiris, V. Pan, S. Pion: “Sign Detection in Residue Number Systems”. *Theoretical Computer Science* **210** (1999), special issue on Real Numbers and Computers, pp. 173–197.

[BFMS00] C. Burnikel, R. Fleischer, K. Mehlhorn, S. Schirra: “A strong and easily computable separation bound for arithmetic expressions involving radicals”. *Algorithmica* **27** (2000) pp. 87–99.

[BFMSS01] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, S. Schmitt: “A separation bound for real algebraic expressions”. *Proc. European Symp. on Algorithms 2001* (ESA 2001), pp. 254–265, Springer LNCS 2161, Berlin, 2001.

[C93] Henri Cohen: *A Course in Computational Algebraic Number Theory*, Springer GTM 138, Berlin, 1993.

[CL$^+$97] David Cox, John Little, Donal O'Shea: *Ideals, Varieties, and Algorithms*, Springer UTM, 2nd ed., New York, 1997.

[DF$^+$00] Olivier Devillers, Alexandra Fronville, Bernard Mourrain, Monique Teillaud: "Algebraic methods and arithmetic filtering for exact predicates on circle arcs". *Proc. 16th Ann. Symp. on Comp. Geom.* (SCG 2000), pp. 139–147, ACM, New York, 2000.

[Dox] D. van Heesch: Doxygen. `http://www.stack.nl/~dimitri/doxygen/`

[ES$^+$02] Arno Eigenwillig, Elmar Schömer, Nicola Wolpert: *Sweeping Arrangements of Cubic Segments Exactly and Efficiently*, ECG-TR-182202-01, INRIA, Sophia-Antipolis, 2002.

[F69] William Fulton: *Algebraic Curves*, Benjamin/Cummings, 1969.
Reprint by Addison-Wesley, 1989.

[GG99] Joachim von zur Gathen, Jürgen Gerhard: *Modern Computer Algebra*, Cambridge Univ. Press, Cambridge, 1999.

[GC$^+$92] K. O. Geddes, S. R. Czapor, G. Labahn: *Algorithms for Computer Algebra*, Kluwer, Norwell/Mass., 1992.

[GH$^+$01] N. Geismann, M. Hemmer, E. Schömer: "Computing a 3-dimensional Cell in an Arrangement of Quadrics: Exactly and Actually!". *Proc. 17th Ann. Symp. on Comp. Geom.* (SCG 2001), pp. 264–273, ACM, New York, 2001.

[G98] C. G. Gibson: *Elementary Geometry of Algebraic Curves: an Undergraduate Introduction*, Cambridge University Press, 1998.

[H97] D. Halperin: "Arrangements". In J. E. Goodman and J. O'Rourke (eds.), *Handbook of Discrete and Computational Geometry*, pp. 389–412, CRC Press, Boca Raton, 1997.

[HW79] G. H. Hardy, E. M. Wright: *An Introduction to the Theory of Numbers*, 5th ed., Oxford University Press, Oxford, 1979.
German translation (3rd ed.): *Einführung in die Zahlentheorie*, Oldenbourg, München, 1958.

[H64] H. Hasse: *Vorlesungen über Zahlentheorie*, 2nd ed., Springer, Berlin, 1964.

[H23] Erich Hecke: *Vorlesungen über die Theorie der algebraischen Zahlen*, Leipzig, 1923. Reprint (with corr.) by Chelsea, New York, 1970.
English translation: *Lectures on the Theory of Algebraic Numbers*, Springer GTM 77, New York, 1981.

[H02] Michael Hemmer: *Reliable computation of planar and spatial arrangements of quadrics*. Master's Thesis, Universität des Saarlandes, Saarbrücken, 2002.

[Hert02] Susan Hert: *Benchmarking of Predicates for Circular Arc Arrangments*, `EXACUS/ConiX/benchmark/using_inria_pred.html`, unpublished, 2002.

[HH$^+$01] S. Hert, M. Hoffmann, L. Kettner, S. Pion, M. Seel: "An Adaptable and Extensible Geometry Kernel". *Proc. 5th Workshop on Algorithm Engineering* (WAE 2001), pp. 76–91, Springer LNCS 2141, Berlin, 2001

[H93] Hoon Hong: “An Efficient Method for Analyzing the Topology of Plane Real Algebraic Curves”. *Proc. IMACS Symp. on Symbolic Computation* (IMACS-SC 93), Lille, 1993.

[HW00] Xiaorong Hou, Dongming Wang: “Subresultants with the Bezout Matrix”. *Proceedings of the Fourth Asian Symp. on Computer Mathematics* (ASCM 2000), pp. 19–28, World Scientific, Singapore New Jersey, 2000.

[KL$^+$99] V. Karamcheti, C. Li, I. Pechtchanski, C. Yap: “A Core Library for Robust Numeric and Geometric Computation”. *Proc. 15th Ann. Symp. on Comp. Geom.* (SCG 99), pp. 351–359, ACM, New York, 1999.

[KC$^+$99] J. Keyser, T. Culver, D. Manocha, S. Krishnan: “MAPC: A library for efficient and exact manipulation of algebraic points and curves”. *Proc. 15th Ann. Symp. on Comp. Geom.* (SCG 99), pp. 360–369, ACM, New York, 1999.

[Kn97] Donald E. Knuth: *The Art of Computer Programming, vol. 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, 1997.

[K95] K. Königsberger: *Analysis 1*, 3rd ed., Springer, Berlin, 1995 (in German).

[K97] K. Königsberger: *Analysis 2*, 2nd ed., Springer, Berlin, 1997 (in German).

[KM0x] W. Krandick, K. Mehlhorn: “On a Theorem of Ostrowski”, in prep.

[L84] S. Lang: *Algebra*, 2nd ed., Addison-Wesley, 1984.

[L85] S. Lang: *Complex Analysis*, 2nd ed., Springer GTM 103, New York, 1985.

[L97] S. Lang: *Undergraduate Analysis*, 2nd ed., Springer UTM, New York, 1997.

[LP$^+$03] Chen Li, Sylvain Pion, Chee K. Yap: *Recent Progress in Exact Geometric Computation*, ECG-TR-243104-01, INRIA, Sophia-Antipolis, 2003.

[L00] Thomas Lücking: *Subresultants*. Master’s Thesis in Mathematics, University of Paderborn, 2000.

[MS$^+$02] R. Martin, H. Shou, I. Voiculescu, A. Bowyer, G. Wang: “Comparison of Interval Methods for Plotting Algebraic Curves”. *Computer Aided Geometric Design* **19(7)** (2002), pp. 553–587.

[M01] Kurt Mehlhorn: *A Remark on the Sign Variation Method for Real Root Isolation*, ECG-TR-123101-01, INRIA, Sophia-Antipolis, 2001.

[MN94] Kurt Mehlhorn, Stefan Näher: *Implementation of a Sweep Line Algorithm for the Straight Line Segment Intersection Problem*, Research Report MPI-I-94-160, Max-Planck-Institut für Informatik, Saarbrücken, 1994.

[MN99] Kurt Mehlhorn, Stefan Näher: *LEDA: A platform for combinatorial and geometric computing*, Cambridge Univ. Press, Cambridge, 1999.

[MS01] Kurt Mehlhorn, Michael Seel: “Infimaximal Frames: A Technique for Making Lines look like Segments”. *Proc. 17th European Workshop on Comp. Geom.* (EUROCG 2001), pp. 78–81, Freie Universität Berlin, 2001.

[M92] M. Mignotte: *Mathematics for Computer Algebra*, Springer, 1992.

[O50] A. M. Ostrowski: "Note on Vincent's Theorem". *Annals of Mathematics, Second Series*, **52.3** (1950), pp. 702–707, 1950.
Reprinted in: Alexander Ostrowski, *Collected Mathematical Papers, vol. 1*, pp. 728-733, Birkhäuser, 1983.

[PS85] F. P. Preparata and M. I. Shamos: *Computational Geometry: An Introduction.* Springer, New York, 1985.

[R92] Reinhold Remmert: *Funktionentheorie 1,* 3rd ed., Springer, Berlin, 1992.

[R01] Günter Rote: "Division-free algorithms for the determinant and the Pfaffian". In H. Alt (ed.), *Computational Discrete Mathematics*, pp. 119–135, Springer LNCS 2122, Berlin, 2001.

[RZ01] F. Rouillier, P. Zimmermann: *Efficient Isolation of Polynomial Real Roots*, Research Report RR-4113, INRIA, France, 2001.

[S91] Takis Sakkalis: "The Topological Configuration of a Real Algebraic Curve". *Bull. Australian Mathematical Society* **43** (1991), pp. 37–50.

[Surf] The *surf* curve and surface plotting software by Stephan Endrass et al. `http://surf.sourceforge.net/`

[T94] G. Taubin: "Rasterizing Algebraic Curves and Surfaces". *IEEE Computer Graphics and Applications* **14** (1994), pp. 14–23.

[We02a] Ron Wein: "High-level filtering for arrangements of conic arcs". *Proc. European Symp. on Algorithms 2002* (ESA 2002), pp. 884–895, Springer LNCS 2461, Berlin, 2002.

[We02b] Ron Wein: *High-level filtering for arrangements of conic arcs*, Master's Thesis, University of Tel Aviv, 2002.

[W02] Nicola Wolpert: *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics*, Ph. D. thesis, Universität des Saarlandes, Saarbrücken, 2002.

[W03] Nicola Wolpert: "Jacobi Curves: The Exact Topology of an Arrangement of Non-Singular Algebraic Curves". *Proc. European Symp. on Algorithms 2003* (ESA 2003), to appear.

[Y00] C. K. Yap: *Fundamental Problems of Algorithmic Algebra*, Oxford University Press, New York, 2000.

[YD95] C. K. Yap, T. Dubé: "The Exact Computation Paradigm". In D.-Z. Du, F. K. Hwang (eds.), *Computing in Euclidean Geometry*, 2nd ed., pp. 452–486. World Scientific, Singapore, 1995.

[YM01] Chee Yap, Kurt Mehlhorn: "Towards Robust Geometric Computation". Talk at the Fundamentals of Computer Science Study Conference, July 2001, Washington DC. `http://www.cs.nyu.edu/exact/talks/focs.html`

A marker like [ABnn, *x.y*] refers to section *x.y* in [ABnn]. References not to sections but pages, theorems, etc. are designated as such.