Exact, Efficient, and Complete Arrangement Computation for Cubic Curves [1]

Arno Eigenwillig [a], Lutz Kettner [a],
Elmar Schömer [b], and Nicola Wolpert [a]

[a] *Max-Planck-Institut für Informatik*
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

[b] *Johannes-Gutenberg-Universität Mainz, Institut für Informatik*
Staudingerweg 9, 55099 Mainz, Germany

Abstract

The Bentley-Ottmann sweep-line method can compute the arrangement of planar curves, provided a number of geometric primitives operating on the curves are available. We discuss the reduction of the primitives to the analysis of curves and curve pairs, and describe efficient realizations of these analyses for planar algebraic curves of degree three or less. We obtain a *complete*, *exact*, and *efficient* algorithm for computing arrangements of cubic curves. Special cases of cubic curves are conics as well as implicitized cubic splines and Bézier curves.

The algorithm is *complete* in that it handles all possible degeneracies such as tangential intersections and singularities. It is *exact* in that it provides the mathematically correct result. It is *efficient* in that it can handle hundreds of curves with a quarter million of segments in the final arrangement. The algorithm has been implemented in C++ as an Exacus library called CubiX.

Key words: Arrangements, algebraic curves, sweep-line algorithm, robustness, exact geometric computation

Email addresses: `arno@mpi-inf.mpg.de` (Arno Eigenwillig), `kettner@mpi-inf.mpg.de` (Lutz Kettner), `schoemer@uni-mainz.de` (Elmar Schömer), `wolpert@mpi-inf.mpg.de` (Nicola Wolpert).

[1] Partially 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).

Article published in Computational Geometry 35 (2006) 36–73

1 Introduction

1.1 Problem Statement

The Bentley-Ottmann sweep-line method [9] can be used to compute the arrangement of planar curves. One *only* has to provide a number of geometric primitives (break a curve into x-monotone pieces, given two curves compute their intersections, compare two intersections or endpoints lexicographically, etc.). The "only" is the crux of the matter; not in principle, but in terms of efficient realization.

We discuss the mathematics of the primitives for cubic curves, i. e., planar algebraic curves of degree three (or less), and derive efficient realizations. Conics as well as implicitized cubic splines and Bézier curves are special cases of cubic curves. We obtain a *complete* (it handles all possible degeneracies), *exact* (it provides the mathematically correct result), and *efficient* (it can handle hundreds of curves with a quarter million of resulting segments) sweep-line algorithm for computing planar arrangements of cubic curves.

With only minor modifications our geometric primitives can also be used to realize incremental approaches. Our implementation can be easily extended to compute arrangements of cubic segments and to perform regularized boolean operations on polygons bounded by them.

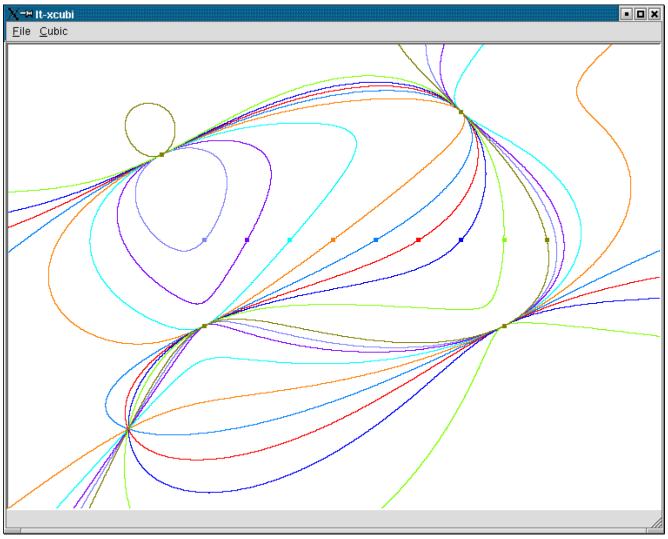

Fig. 1. A screenshot of our implementation showing a pencil of nine cubics with four common tangential intersections.

Complete, exact, and efficient implementations for planar arrangements of straight line segments exist, e. g., in LEDA [62, ch. 10.7], and in the planar map [34] and planar Nef-polyhedron [74] classes of CGAL. However, existing implementations for curved objects are either incomplete, inexact, or not aimed at efficiency except for some recent work on circle and conic arcs (see below). For cubic curves we are not aware of any complete, exact, and efficient implementation.

Our work is influenced by results from three different communities: computer aided geometric design (CAGD), computational geometry, and computer algebra. The problem of computing intersections of curves and surfaces has a long history in CAGD. The CAGD community concentrates on approximate solutions by numerical methods which cannot distinguish, e. g., between a tangential intersection and two intersections lying very close together. Complete and exact implementations have been addressed recently: MAPC [50] is a library for exact computation and manipulation of algebraic curves. It offers arrangements of planar curves but does not handle all degenerate situations. Yap [82] uses separation bounds to make the traditional subdivision algorithm for the intersection of Bézier curves exact and complete in the presence of tangential and almost-tangential intersections.

Arrangements, mostly of linear objects, are also a major focus in computational geometry; see the survey articles of Halperin [43] and Agarwal/Sharir [2]. Many exact methods for curved objects have been formulated for the Real RAM model of computation [67], which allows unit-time operations on arbitrary real numbers and conceals the high cost of exact arithmetic with algebraic numbers.

Sakkalis [70,71], Hong [45], and Gonzalez-Vega/Necula [42] analyze the topology of a single real algebraic curve. Their approaches, like ours, can be seen as special forms of Cylindrical Algebraic Decomposition [23,5]. However, they do not consider the interaction between pairs of curves, and the full algebraic machinery they deploy is not necessary for cubic curves. The recent textbook by Basu, Pollack, and Roy [8] is a very useful reference for the algebraic background of those methods and also contains a curve topology algorithm. Aspects of the crucial problem to capture behavior at irrational points by rational arithmetic were treated by Canny [20] (Gap Theorem) and Pedersen [66] (multivariate Sturm sequences).

Predicates for arrangements of circular arcs that reduce all computations to sign determination of polynomial expressions in the input data are treated by Devillers et al. [26]. Recent work by Emiris et al. [31,32] discusses some predicates on conics in this style. However, these approaches do not extend easily to more complicated curves.

Exact, efficient, and complete algorithms for planar arrangements have been published by Wein [77] and Berberich et al. [11] for conic segments, and by Wolpert [78,73] for special quartic curves as part of a surface intersection algorithm. A generalization of Jacobi curves (used for locating tangential intersections) is described by Wolpert [79].

The work presented here follows the Master's thesis of the first author [28] and has appeared as an extended abstract in [29].

1.3 Our Results

What are the difficulties in going from straight lines and straight line segments to conics and further on to cubic curves? The defining equations and thus the geometry of the basic objects and the coordinates of their intersections become more complicated. For straight line segments with rational endpoints, all vertices in their arrangement have rational coordinates. In the case of curves, the intersections are solutions to systems of non-linear polynomial equations and thus, in general, irrational algebraic numbers. We review polynomials, algebraic numbers, and elimination of variables among polynomial equations in Section 2.

The sweep-line algorithm works on x-monotone segments. Lines and line segments are x-monotone, conics need to be split at points of vertical tangent, and cubic curves need to be split at points of vertical tangent and at singularities. These notions and other required elements of the geometry of algebraic curves are introduced in Section 3. Cubic curves have a more diverse geometry than conics. Its analysis is a separate step in our algorithm, explained in Section 4.1.

We turn to pairs of curves. Two lines intersect in a single point with rational coordinates. Two conics intersect in up to four points. Each of the intersecting conic arcs can be parametrized in the form $y(x)$ using a single square root. In the case of a tangential intersection, the x-coordinate of the intersection can also be written as an expression involving a single square root. Therefore, previous work on conics [77,11] could make heavy use of the existing efficient methods for arithmetic within FRE [48,18], the field of real root expressions. FRE is the closure of the integers under the operations $+$, $-$, $*$, $/$, and $\sqrt[k]{\ }$ for arbitrary but fixed k; it is a subfield of the real algebraic numbers. However, the $\sqrt[k]{\ }$ operation is restricted to extracting real-valued roots, thus general polynomial equations of degree ≥ 3 are not solvable in FRE, severely limiting its applicability to cubics. (The usability of recent additions for arithmetic with all real algebraic numbers [55,19,72] remains to be investigated.) Hence we need more powerful techniques. We discuss them and the geometric analysis of curve pairs in Section 4.2.

The central idea of our approach to curve and curve pair analyses is to exploit geometric properties of cubic curves in order to avoid arithmetic with irrational numbers as far as possible.

In Section 5 we put everything together and discuss high-level issues of the Bentley-Ottmann sweep for algebraic curves and how the required predicates reduce to curve and curve pair analyses. This part applies to algebraic curves of any degree.

We conclude with a discussion of running time from a theoretical (Section 6) and especially an experimental (Section 7) point of view. We point out that there is a full implementation of our algorithm.

2 Algebraic Foundations

The algorithmic handling of algebraic curves rests to a large extent on algebraic operations. We give a concise summary of the existing results we use, just enough to make our presentation self-contained and accessible to non-experts in symbolic computation. This summary also provides the necessary context for describing the implementation choices we made.

As reference for the abstract algebra involved here we mention the textbook by Lang [54]. The algorithmic aspects are covered in the computer algebra books by Geddes/Czapor/Labahn [38], Akritas [4], Cohen [21], and von zur Gathen/Gerhard [36]. The recent book by Basu/Pollack/Roy [8] combines the viewpoints of real algebraic geometry and computer algebra.

2.1 Rings and Fields

Recall the algebraic notions of ring and field. All fields considered in the sequel have characteristic zero, i. e., contain the rational numbers $\mathbb{Q}$. A *ring* in modern terminology is a commutative ring with unity. We additionally demand it to contain the integers $\mathbb{Z}$, and unless specifically stated otherwise, to be free of zero divisors. With our terminology, 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, $\mathrm{Q}(\mathbb{Z}) = \mathbb{Q}$.

We say d *divides* r and write $d|r$ if there is $c \in R$ such that $r = cd$. If $u|1$,

u is a *unit* of R. A non-unit $r \neq 0$ that cannot be written as product of two other non-unit elements is called an *irreducible* element. A ring R is a *unique factorization domain* (UFD) if any non-zero non-unit element r possesses a factorization

$$r = \prod_{i=1}^{k} p_i \tag{1}$$

into irreducible elements p_i that is unique up to reordering the p_i and multiplying them by units. In any UFD R, the irreducible elements p are *prime*, meaning that for all $r, s \in R$ we have $p|rs \;\Rightarrow\; p|r \vee p|s$. (For a proof, see [54, II.4].) Given $r \in R$ and some irreducible element p, the maximal exponent $e \in \mathbb{N}_0$ such that $p^e|r$ is called the *multiplicity* of p in r.

2.2 Polynomials

2.2.1 Basics

Adjoining an indeterminate x to a ring R yields the *univariate* polynomial ring $R[x]$. We call $f \in R[x]$ *monic* if its leading coefficient $\ell(f)$ is 1. Iterated adjunction of indeterminates leads to *multivariate* polynomials $R[x_1]\cdots[x_n] = R[x_1, \ldots, x_n]$. The units of $R[x_1, \ldots, x_n]$ are the units of R.

A multivariate polynomial can be seen *recursively* as a univariate polynomial in x_n with coefficients in $R[x_1]\cdots[x_{n-1}]$, or *flat* as a sum of terms $a_{i_1 \ldots i_n} x_1^{i_1} \cdots x_n^{i_n}$. Its *total degree* $\deg(f)$ is $\max\{i_1 + \ldots + i_n \mid a_{i_1 \ldots i_n} \neq 0\}$.

A polynomial is *homogeneous* if all its terms have the same total degree. Otherwise, it decomposes uniquely as a sum of its *homogeneous parts* $f = f_0 + f_1 + \ldots + f_{\deg(f)}$ where f_d is homogeneous of degree d. The d-th order terms f_d are called *constant*, *linear*, *quadratic*, etc., *part* of f, for $d = 0, 1, 2, \ldots$, respectively.

Let $k \leq n$ and $\xi \in R^k$. Substituting values $\xi_1, \ldots, \xi_k$ for $x_1, \ldots, x_k$ homomorphically maps $f \mapsto f|_\xi = f(\xi_1, \ldots, \xi_k, x_{k+1}, \ldots, x_n) \in R[x_{k+1}, \ldots, x_n]$.

A polynomial $f \in R[x_1, \ldots, x_n]$ is called *x_n-regular* if it contains a term $cx_n^{\deg(f)}$ with $0 \neq c \in R$. Its degree does not change when substituting values for $x_1, \ldots, x_k$, $k < n$. The product of two x_n-regular polynomials is x_n-regular. Any factor of an x_n-regular polynomial is x_n-regular.

2.2.2 Unique factorization and the GCD

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

$$f = qg + r, \quad \deg(r) < \deg(g). \tag{2}$$

The constructive proof of this proposition yields a simple and reasonably efficient algorithm to compute these quantities. Iff $g|f$, then it produces the remainder 0 and the quotient f/g.

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 $d' \in R$. To compute a gcd in $K[x]$, there is the *Euclidean Algorithm*: Given two non-zero polynomials $f, g \in K[x]$, repeatedly replace one of larger (or equal) degree with its remainder modulo the other one, until zero occurs as remainder. The last non-zero remainder d is $\gcd(f, g)$. By accumulating intermediate results, the *Extended Euclidean Algorithm* (EEA) additionally computes *Bézout factors* $p, q \in K[x]$ with the property $d = pf + qg$. See [38, 2.4], [36, 3.2], [80, 2.2] or [21, 3.2.1].

The EEA demonstrates that $K[x]$ is a principal ideal domain and hence a UFD [54, II.4, V.4]. Thus any non-zero polynomial $f \in K[x]$ has a unique (up to order) factorization

$$f = \ell(f) \cdot \prod_{j=1}^{k} q_j^{e_j} \tag{3}$$

into its leading coefficient and powers of pairwise distinct monic irreducible factors q_j with their respective multiplicities $e_j > 0$. This follows from (1) by normalizing and grouping factors.

Let us now consider the divisibility properties of polynomials over a UFD R. In this setting, non-unit constant factors are relevant. For a non-zero polynomial $f \in R[x]$, define the *content* $\operatorname{cont}(f)$ of f as the gcd of its coefficients. If $\operatorname{cont}(f) = 1$, then f is called *primitive.*

Proposition 1 (Gauss' Lemma)
Let R be a UFD. Let $f, g \in R[x]$ be non-zero. Then $\operatorname{cont}(fg) = \operatorname{cont}(f)\operatorname{cont}(g)$.

For a proof, see [54, V.6]. The following easy corollary relates the divisibility properties of $R[x]$ and $\mathrm{Q}(R)[x]$.

Corollary 2 *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.

Continuing that line of thought [54, V.6], one obtains:

Theorem 3 (Gauss' Theorem)
If R is a UFD, then $R[x_1, \dots, x_n]$ is a UFD.
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 essentially uniquely into coprime irreducible factors with their respective multiplicities. (We shall meet such irreducible factors again later on for $n = 2$ as components of curves.) It follows that there exists $\gcd(f, g)$ for non-zero $f, g \in K[x_1, \dots, x_n]$. The following easy corollary to Proposition 1 is useful for its computation.

Corollary 4 *Let R be a UFD, and let $f, g \in R[x]$ be non-zero polynomials. Then* $\operatorname{cont}(\gcd(f, g)) = \gcd(\operatorname{cont}(f), \operatorname{cont}(g))$.

Suppose we can compute gcds in a UFD R. Then we can also compute $\gcd(f, g)$ for non-zero $f, g \in R[x]$: Because of Corollary 2, $\gcd(f, g)$ is, up to a constant factor, the gcd of f, g regarded as elements of $\mathrm{Q}(R)[x]$, and we can compute that with the Euclidean Algorithm. Using Corollary 4, the error in the constant factor can then be corrected by adjusting the content to $\gcd(\operatorname{cont}(f), \operatorname{cont}(g))$. This allows us to compute gcds, for example, in $\mathbb{Z}[x]$; and, by a recursive application of this approach, in $R[x_1, \dots, x_n]$ or $K[x_1, \dots, x_n]$. We refer to [38] for details.

An efficient implementation of this scheme employs $\mathrm{Q}(R)$ only conceptually and keeps all coefficients in R in order to avoid costly fractional arithmetic. To obtain (constant multiples of) remainders in $R[x]$, one changes Euclidean division into *pseudo division* by replacing f with $\ell(g)^{\deg(f)-\deg(g)+1} f$ in (2), so that all divisions of coefficients become possible within R. To curb coefficient growth, one needs to predict a large constant factor of each remainder and divide it out. To do this, we use the subresultant method due to Collins [22] and Brown/Traub [17]. It is described by Loos [59, 4.5], Brown [16], and Knuth [51, pp. 428+]. (A subsequent improvement has been obtained by Lickteig and Roy [56, §4] [57, Thm. 4.1], see also [58], [8, 8.3] or [30].)

2.2.3 Multiplicities and Derivatives

Whenever we have unique factorization into irreducibles, we can formulate the weaker notion of *square-free factorization*, or *factorization by multiplicities* as we like to call it. Take (3) and group factors q_i with equal multiplicities $m = e_i$ into one factor s_m:

$$f = \ell(f) \cdot \prod_{m=1}^{\max_i e_i} s_m^m, \qquad \text{where } s_m = \prod_{e_i = m} p_i \tag{4}$$

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 non-unit factor in common. Define the *square-free part* of f as $\prod_m s_m$.

The significance of this weaker form of polynomial factorization lies in the relative ease of computing it, using derivatives. For a polynomial $f = \sum_{i=0}^{n} a_i x^i$ we define its *(formal) derivative* as $f' := \sum_{i=1}^{n} i a_i x^{i-1}$. For multivariate polynomials $f \in R[x_1, \ldots, x_n]$, partial derivatives are defined accordingly and denoted by subscripts like $f_{x_1 x_2}$. Partial differentiation of polynomials w. r. t. different variables commutes. The result of differentiating f exactly $r \geq 0$ 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* $\nabla f := (f_{x_1}, \ldots, f_{x_m})^{\mathsf{T}}$ of f.

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

Proposition 5 *Let K be a field, let $f \in K[x]$ be a non-zero polynomial, and let $f = \ell(f) \prod_{m=1}^{M} s_m^m$ be its square-free factorization. Then*

$$\gcd(f, f', \ldots, f^{(k)}) = \prod_{m=k+1}^{M} s_m^{m-k} \quad \textit{(up to units)} \tag{5}$$

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 is performed by applying the following observation separately to each irreducible factor of f: 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-free factorization, like gcd computation, does not depend on the choice of K and applies equally to all fields containing the coefficients of f. Furthermore, the statement of the proposition remains valid for an x_n-regular polynomial $f \in K[x_1, \ldots, x_n]$ when derivatives are taken w. r. t. x_n.

The relation (5) of multiplicities and derivatives allows us to compute a square-free factorization of $f \in K[x]$ or of $f \in R[x]$ for a UFD R by repeated differentiation and gcd computation. Yun's Algorithm [38, 8.2] does this in a clever way to keep the coefficients of intermediate polynomials small.

This algorithm prompts an implementation choice concerning the computation of *gcd-free parts*; that is, passing from a pair (f, g) of polynomials to the triple $(f/d, g/d, d)$ where $d = \gcd(f, g)$. We choose the obvious implementation (compute d and divide by it). The alternative of executing the Extended Euclidean Algorithm and obtaining the gcd-free parts as cofactors of the zero polynomial at the end of the polynomial remainder sequence [8, Thm. 10.13] did not seem to offer a better performance for the univariate instances of the problem met in our application.

2.2.4 Roots of Polynomials

Let K be a field, and let $f \in K[x]$. As a consequence of division with remainder, we have that $f(\xi) = 0$ 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. Furthermore, it allows us to conclude that a polynomial f 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.

A field K is called *algebraically closed* if the second alternative does not occur. For every field K there exists an essentially unique $\subseteq$-smallest algebraically closed field $\overline{K}$ containing K, called the *algebraic closure* of K. Each element of $\overline{K}$ is a root of some polynomial with coefficients in K. See [54, VII.2]. The algebraic closure of the reals $\mathbb{R}$ is the field $\mathbb{C} = \mathbb{R}(i)$ of complex numbers. As a proper subset, it contains the algebraic closure $\overline{\mathbb{Q}}$ of the rationals $\mathbb{Q}$.

Let ϑ be an algebraic number. $\mathbb{Q}(\vartheta)$ denotes the $\subseteq$-minimal subfield of $\overline{\mathbb{Q}}$ containing ϑ. Among all monic polynomials from $\mathbb{Q}[x]$ that vanish at ϑ, there is one of minimal degree. It is uniquely determined and called the *minimal polynomial* f of ϑ. The *degree* of ϑ is defined as $\deg(f)$. The kernel of the evaluation homomorphism $\mathbb{Q}[x] \to \mathbb{Q}(\vartheta)$, $x \mapsto \vartheta$, consists precisely of the multiples of f, so that the field $\mathbb{Q}(\vartheta)$ is isomorphic to the quotient ring $\mathbb{Q}[x]/(f)$, see [54, II.1, VII.1]. Now consider all roots $\vartheta_1 := \vartheta, \vartheta_2, \ldots, \vartheta_d$ of f. They are algebraically indistinguishable insofar as $\mathbb{Q}(\vartheta_i)$ and $\mathbb{Q}(\vartheta_j)$ are isomorphic for any $1 \leq i, j \leq d$; hence a polynomial equation with rational coefficients is satisfied by none or by all of the ϑ_i. We call the ϑ_i the *algebraic conjugates* of ϑ. Observe the analogy with the usual complex conjugation defined by replacing the imaginary unit i with the other root $-i$ of its minimal polynomial $x^2 + 1$.

The notion of algebraic conjugacy extends to n-tuples ξ of algebraic numbers by choosing an algebraic number ϑ such that all ξ_i are rational expressions in ϑ, and ϑ is conversely a rational expression in $\xi_1, \ldots, \xi_n$. (Such ϑ exists and is called the *primitive element* of $\mathbb{Q}(\xi_1) \cdots (\xi_n)$, see [54, VII.6] for an abstract and [60, Thm. 11] for a constructive proof.) The conjugates of ξ are the n-tuples obtained by replacing ϑ with its conjugates. No two distinct conjugates of ϑ produce the same ξ. If ξ is a solution to a system of polynomial equations with rational coefficients, then any of the $\deg(\vartheta)$ many conjugates is a solution as well.

2.3 Resultants

Intersecting two algebraic curves f and g essentially means solving a system $f(x, y) = g(x, y) = 0$ of two polynomial equations. For this purpose, we will need some results from elimination theory.

2.3.1 The Sylvester Resultant

We start by looking at the solvability of a system of two equations in *one* variable.

Proposition 6 *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 $(n + m) \times (n + m)$ Sylvester matrix*

$$\mathrm{Syl}(f, g) = \begin{pmatrix} a_m & \cdots & \cdots & a_0 & & \\ & \ddots & & & \ddots & \\ & & a_m & \cdots & \cdots & a_0 \\ b_n & \cdots & \cdots & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_n & \cdots & \cdots & b_0 \end{pmatrix} \tag{6}$$

vanishes.

Proof. The equivalence of (i)–(iii) is easy. The equivalence of (iii) and (iv) follows by linear algebra, noting that the transpose $\mathrm{Syl}(f, g)^{\mathsf{T}}$ 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. See [25, 3.5] for more details. □

The determinant $\det(\mathrm{Syl}(f, g))$ is called the *resultant* $\mathrm{res}(f, g, x)$ of f and g with respect to x. The following proposition is not surprising in the light of (i) $\Leftrightarrow$ (iv):

Proposition 7 *Let K be a field. Consider the two non-zero polynomials $f = a_m \prod_{i=1}^{m}(x - \alpha_i)$ and $g = b_n \prod_{j=1}^{n}(x - \beta_j)$ in $K[x]$. It holds that*

$$\operatorname{res}(f, g, x) = a_m^n b_n^m \prod_{i=1}^{m} \prod_{j=1}^{n} (\alpha_i - \beta_j). \tag{7}$$

Proofs can be found in [54, V.10] [60, Thm. 1] [21, 3.3.2].

Proposition 6 can be read as a statement on the Euclidean Algorithm executed for f and g: Iff $\det(\operatorname{Syl}(f, g))$ vanishes, the polynomial remainder sequence of f and g ends with zero before a constant gcd is reached. In fact, there is a deep connection between subdeterminants of the Sylvester matrix and polynomial remainder sequences; see [59], [80, 3.6–3.9] or [8, 8.3]. We state only its most prominent part, using the following notion:

Let K be a field, and let $f, g \in K[x]$ be two non-zero polynomials with degrees higher than $k \geq 0$. Then the k-th *subresultant*[2] $\operatorname{sres}_k(f, g, x)$ of f and g is defined as the determinant of the matrix obtained from $\operatorname{Syl}(f, g)$ by deleting the last $2k$ columns of the matrix and the last k rows of each of the two parallelograms of coefficients.

Proposition 8 *With f, g and k as above, the remainder sequence computed by the Euclidean algorithm for f and g contains a remainder of degree k iff the k-th subresultant of f and g does not vanish. In particular,*

$$\operatorname{sres}_i(f, g, x) = 0 \textit{ for all } 0 \leq i < k \iff \deg(\gcd(f, g)) \geq k. \tag{8}$$

A proof appears in [36, 6.10]. In the sequel, we will only need (8) specialized to $k = 2$; for this a direct proof is given in [4, Thm. 5.2.5].

2.3.2 Elimination with the Resultant

Now consider bivariate polynomials $f, g \in K[x, y]$. Proposition 6 can help us to solve $f = g = 0$ when we view f and g as polynomials in $Q(K[x])[y]$. The solution proceeds in two stages [25, 3.1]:

(1) The *projection step* in which we determine the *partial solutions* ξ, i. e., the values of x for which the system $f = g = 0$ permits a solution (ξ, η).

[2] The literature is divided between two definitions of subresultant. This article uses the notion of a *scalar subresultant*. In the context of *polynomial subresultants*, these are called *principal subresultant coefficients*. See [37] for a systematic comparison. See [59], [80, 3.6–3.9], [8, 8.3] and [30] for the theory of polynomial subresultants.

(2) The *extension step* in which we extend a partial solution $x = \xi$ by all possible values η of y such that $f(\xi, \eta) = g(\xi, \eta) = 0$.

In this setting, $\operatorname{res}(f, g, y)$ is a polynomial from $K[x]$, 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 both $\ell(f)$ and $\ell(g)$. Then the Sylvester matrix $\operatorname{Syl}(f, g)|_\xi$ evaluated at ξ has a leftmost column full of zeroes, and so $\operatorname{res}(f, g, y)(\xi) = 0$ 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. However, if one of the polynomials is y-regular, everything is fine:

Proposition 9 *Let K be a field, and let $f, g \in K[x, y]$ be non-zero polynomials. Furthermore, let f be y-regular. Then for all $\xi \in K$ the two conditions*

(i) $\operatorname{res}(f, g, y)(\xi) = 0$

(ii) There is $\eta \in \overline{K}$ such that $f(\xi, \eta) = g(\xi, \eta) = 0$

are equivalent.

Proof. We show that $\operatorname{res}(f, g, y)(\xi) = 0$ is equivalent to $\operatorname{res}(f|_\xi, g|_\xi, y) = 0$: Observe that $\operatorname{Syl}(f, g)|_\xi$ contains $\operatorname{Syl}(f|_\xi, g|_\xi)$ as a lower right submatrix S. The $d := \deg(g) - \deg(g|_\xi) \geq 0$ columns left of S are zero below the diagonal and contain the constant diagonal entries $\ell(f) \neq 0$. Thus $\det(\operatorname{Syl}(f, g)|_\xi) = \ell(f)^d \det(\operatorname{Syl}(f|_\xi, g|_\xi))$, proving the claim. □

This proposition extends in the obvious fashion to subresultants and their property from Proposition 8.

When intersecting two curves, we aim for finitely many intersection points, so we want to obtain a resultant which is not the zero polynomial.

Proposition 10 *Let K be a field, and let $f, g \in K[x, y]$ be non-zero polynomials. Then* $\operatorname{res}(f, g, y) \neq 0$ *iff f and g have no common factor of positive degree in y.*

Proof. Recall from §2.2.2 that $K[x]$ is a UFD. Hence Corollary 2 implies that f and g have no common factor of positive univariate degree as elements of $K[x][y]$ iff they have no common factor of positive univariate degree as elements of $\mathrm{Q}(K[x])[y]$. By Proposition 6, this is in turn equivalent to $\operatorname{res}(f, g, y) \neq 0$. □

Another practical concern relates to computations over the field of real numbers $\mathbb{R}$ which is not algebraically closed. A real zero of $\operatorname{res}(f, g, y)$ may arise from a complex solution whose x-coordinate happens to be real. However, complex solutions come in pairs of complex conjugates. We will later guarantee uniqueness of solutions extending a given x-coordinate, so that real zeroes of $\operatorname{res}(f, g, y)$ are certain to come from real solutions.

2.3.3 Computing the Resultant

We have considered three options for computing the resultant of two polynomials f and g (up to sign):

(1) Compute the determinant of the Sylvester matrix $\operatorname{Syl}(f, g)$ defined above. This is the most obvious approach.
(2) Compute the determinant of the Bézout matrix $\operatorname{Bez}(f, g)$ defined below. This determinant is smaller and hence faster to compute.
(3) Compute a polynomial remainder sequence of f and g with the subresultant algorithm mentioned in §2.2.2. It computes subresultants incrementally along with the remainder sequence and uses them to curb coefficient growth; in particular, it computes the zeroth subresultant, which is the resultant.

In computing the determinant of a matrix with entries from a ring (in our case: integers or polynomials), it is preferable to avoid arithmetic with fractions for the sake of efficiency. We have considered two choices for fraction-free determinant computations, namely, the algorithms of Gauss-Bareiss [21, 2.2.3] and Berkowitz[3] [68]. Gauss-Bareiss needs $O(n^3)$ arithmetic operations, including divisions that are possible within the ring of matrix entries; whereas Berkowitz needs $O(n^4)$ arithmetic operations, none of them divisions. (Here, n is the row and column dimension of the matrix.)

Our main application of resultants concerns y-regular bivariate polynomials $f, g \in \mathbb{Z}[x][y]$ (viewed recursively) of degree at most 3. For this specific setting, with our implementations of the various alternatives, and with our benchmark data, we have observed that Berkowitz' method for determinant computation is faster than Gauss-Bareiss, despite the asymptotics; and that evaluating the Bézout determinant with Berkowitz is faster than using the Sylvester determinant or a subresultant remainder sequence. This is consistent with the results of a recent study by Abdeljaoued et al. [1] for polynomials with higher degrees and more inner variables.

[3] This naming follows Rote [68, 2.8]. Others have attributed this algorithm to Samuelson.

Hence let us take a look at the Bézout matrix of two polynomials $f, g \in K[x]$ both having degree m. Consider the polynomial $f(x)g(z) - f(z)g(x)$ involving the new indeterminate z. It vanishes whenever $x = z$ and therefore is divisible by $x - z$. The quotient is

$$\Lambda(x, z) := \frac{f(x)g(z) - f(z)g(x)}{x - z} = \sum_{i,j=0}^{m-1} c_{ij}\, x^i z^j. \tag{9}$$

The *Bézout matrix* of f and g is the $m \times m$ coefficient matrix $\mathrm{Bez}(f, g) := (c_{ij})_{i,j=0}^{m-1}$ of Λ. The book by Gelfand/Kapranov/Zelevinsky [39, 12.1] records the following classical results:

Proposition 11 *Let K be a field. Let $f = \sum_{i=0}^{m} a_i x^i$ and $g = \sum_{i=0}^{n} b_i x^i$ be two polynomials in $K[x]$ of degrees m and n, resp., where $m \geq n$.*
The entries of the Bézout matrix $\mathrm{Bez}(f, g) = (c_{ij})_{i,j=0}^{m-1}$ are given by

$$c_{ij} = \sum_{k=0}^{\min(i,j)} [k,\ i + j + 1 - k], \quad \textit{where } [i, j] := a_i b_j - a_j b_i, \tag{10}$$

with the convention that $a_i = 0$ for $i > m$ and $b_i = 0$ for $i > n$.
The determinant of the Bézout matrix is

$$\det(\mathrm{Bez}(f, g)) = \pm a_m^{m-n} \mathrm{res}(f, g, x). \tag{11}$$

For $m > n$, the extraneous power of a_m in (11) can be avoided by forming a *hybrid Bézout matrix* of size $m \times m$ that consists of n Bézout-like rows and $m - n$ Sylvester-like rows [27] [46]. We compute the resultant of f and g as determinant of this hybrid Bézout matrix using Berkowitz' method.

The k-th subresultant can be expressed as an $(m - k) \times (m - k)$ subdeterminant of the (hybrid) Bézout matrix [27] [46]. This is how we compute subresultants.[4]

We shall give two further references: The (sub)resultant property of (sub)determinants of the Bézout matrix is derived from first principles (i. e., without reference to the Sylvester matrix) by Goldman et al. [41]. The general Bézout construction of resultants to eliminate several variables at once is treated extensively by Bikker and Uteshev [12], including an explicit discussion of the subresultant properties in the univariate case considered here.[5]

[4] The recent article [1] explains how to obtain all subresultants from one execution of Berkowitz' method. While quite interesting in general, this would not gain a lot in our setting where $m \leq 3$, $k \leq 1$.

[5] Be warned that the matrix $\mathbf{B}$ considered in [12] differs from our *Bezoutiant* $\mathrm{Bez}(f, g)$ by a change of basis.

2.4 Finding and Handling Roots of Polynomials

2.4.1 Root Isolation

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 $\{x \in [l, r] \mid f(x) = 0\} = \{\xi\}$.

Before anything else, we make the coefficients of f integral and factor f by multiplicities (see §2.2.3). Then we determine the zeroes of each square-free factor separately.

We shortcut the factorization by multiplicities if $\operatorname{res}(f, f', x) \not\equiv 0$ modulo some prime, because $\operatorname{res}(f, f', x) \neq 0$ is equivalent to f being already square free (see §2.2.3 and §2.3.1). Checking this condition modulo a prime small enough to allow arithmetic in a `double`'s mantissa (following [15]) is much faster than computing $\gcd(f, f') = 1$ from a polynomial remainder sequence over the integers. In analogy to floating-point filters, we call this a *modular filter*. Squarefreeness is the expected case in our uses of root finding. The modular filter allows us to handle this case efficiently with high probability, and it is simpler to implement than a full-blown modular gcd algorithm.

After factorization, we process each square-free factor $\sum_{i=0}^{n} a_i x^i$ separately. We compute an interval $[-B, B]$ enclosing all its real roots as follows: Determine

$$k_0 := \min \left\{ k \geq 0 \;\middle|\; |a_n| \, 2^{nk} > \sum_{i=0}^{n-1} |a_i| \, 2^{ik} \right\} \tag{12}$$

by successively trying $k = 0, 1, 2, 3, \ldots$; then take $B = 2^{k_0}$ (see [63, p. 144]).

To perform the actual root isolation, we use the Descartes Method, [6] a divide-and-conquer approach based on Descartes' Rule of Signs. Its modern form goes back to Collins and Akritas [24], see also Krandick [52] and Rouilliler/Zimmermann [69].

Proposition 12 (Descartes' Rule of Signs)
Let $f \in \mathbb{R}[x]$ be a non-zero polynomial with exactly p positive real roots (counted with multiplicities) and exactly v sign variations in its coefficient sequence. Then $v - p$ is non-negative and even.

Proofs can be found in [4, Thm. 7.2.6], [63, Thm. 5.5], and [53]. A *sign variation* in a sequence $a_0, \ldots, a_n$ of real numbers is a pair of indices $0 \leq i < j \leq n$

[6] The designation "Uspensky's Method" is common but incorrect [3].

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.

If $v = 0$ or $v = 1$, then v is known to be the exact number of positive real roots. If $v > 1$, this count might include pairs of complex-conjugate roots close to the positive real axis.

Descartes' Rule 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 applying it to the composition $f \circ T$ with a Möbius transformation $T(x) = \frac{ax+b}{cx+d}$ taking $]0, +\infty[$ to $]l, r[$, see [4, 7.3.2]. (A reversed bracket indicates exclusion of the boundary point.)

The Descartes Method works as follows: Starting from the initial interval enclosing all roots, recursively subdivide intervals at their midpoint until each interval under consideration is known to be isolating ($v = 1$) or empty ($v = 0$). The crux is to prove that $v \in \{0, 1\}$ will hold eventually. The oldest proof is from A. J. H. Vincent (1836). We refer the reader to [53] for a rediscovered result of Ostrowski [65] that gives a particularly strong partial converse.

An alternative to the Descartes Method is provided by *Sturm Sequences*. These are polynomial remainder sequences of f and f' with special arrangements for signs of the remainders (see [4, 7.2.2], [63, 5.4.1] or [80, chap. 7]), so unless the modular filter lets us skip that step, we essentially do compute a Sturm sequence for f during its square-free factorization. Nevertheless, we do not use it for root isolation. An empirical study by Johnson [47] indicates that root isolation based on Sturm Sequences is often inferior to the Descartes Method, even if the cost of obtaining the Sturm sequence is excluded.

2.4.2 Representing and ordering real algebraic numbers

Let the polynomial $f_0 \in \mathbb{Q}[x]$ have a real root $\vartheta \in \mathbb{R}$. The procedure of §2.4.1 computes a representation $\vartheta \mathrel{\widehat{=}} (f, [l, r])$ for it consisting of a square-free factor $f \in \mathbb{Z}[x]$ of f_0 and an isolating interval with boundaries $l, r \in \mathbb{Q}$. (In the field of symbolic computation, this representation is standard [60].) Except for the degenerate case $l = r = \vartheta$, we make sure $\vartheta \in]l, r[$.

We want to compare numbers of that form. A key step is the bisection of a non-degenerate isolating interval $[l, r]$ at a point $t \in]l, r[$. 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 $[l, r]$ 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 $\vartheta_1 \mathrel{\widehat{=}} (f_1, [l_1, r_1])$ and $\vartheta_2 \mathrel{\widehat{=}} (f_2, [l_2, r_2])$ be two algebraic numbers which 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 $[l, r] := [l_1, r_1] = [l_2, r_2]$ with $l < r$. The polynomials f_1 and f_2 are square free and each has exactly one zero in $[l, r]$. Consider $d := \gcd(f, g)$, which is also square free. If $\operatorname{sign}(d(l)) \neq \operatorname{sign}(d(r))$, there is a common zero $\vartheta_1 = \vartheta_2$ in $[l, r]$ and we are done. Otherwise, d has no zero in $[l, r]$, hence $\vartheta_1 \neq \vartheta_2$, 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.

After a comparison of ϑ_1 and ϑ_2, the isolating intervals in their representations are replaced by their refinements. If we have computed a non-constant gcd d, we also replace each f_i by d or f_i/d, respectively, whichever vanishes at ϑ_i, as is discernible from the sign change.

As before in §2.4.1, the idea of a *modular filter* is applicable. When comparing two algebraic numbers ϑ_1 and ϑ_2 of unrelated origin, the expected case is that their defining polynomials are coprime. Hence before computing $d := \gcd(f_1, f_2)$, our implementation checks whether $\operatorname{res}(f_1, f_2, x) \not\equiv 0$ modulo some prime (see §2.3.1). If this holds, we know $d = 1$. More on such optimizations can be found in [44].

A variation of the method above allows to decide for $\vartheta \mathrel{\widehat{=}} (f, [l, r])$ whether $g(\vartheta) = 0$ or not: Compute $d := \gcd(f, g)$ and decide (by inspecting signs at l and r) whether d vanishes at ϑ or not.

This representation of real algebraic numbers can be seen as a delayed form of numerical root finding with a portion of symbolic computation to handle equality accurately. Subsequent comparisons benefit from the refinements done earlier. Past inequality results are cached in the form of non-overlapping isolating intervals. The refinement of interval boundaries in a comparison of unequal numbers $\vartheta_1 < \vartheta_2$ computes a rational number $r \in]\vartheta_1, \vartheta_2[\cap \mathbb{Q}$ along the way.

One can implement a union-find scheme to cache equality. This causes all logical consequences of transitivity, reflexivity and symmetry of $=$ to be evident from unitedness. When uniting $\vartheta_1 = \vartheta_2$, replace the isolating intervals $[l_1, r_1], [l_2, r_2]$ by their intersection. All logical consequences of transitivity of $>$ previously reflected by $[l_1, r_1]$ or $[l_2, r_2]$ are also reflected by their intersection. Thus, any logical consequence of a sequence of three-valued comparisons of algebraic numbers with union can afterwards be read off without further interval refinements.

We will need one more operation on algebraic numbers, namely, refining the isolating interval $[l, r]$, $l \neq r$, of a zero ϑ of f against all zeroes of another square-free polynomial $g \in \mathbb{Z}[x]$ different from ϑ. To achieve this, refine $[l, r]$ by bisection until one of two conditions holds: Either Descartes' Rule of Signs implies the absence of a zero of g in the interval; or Descartes' Rule of Signs implies the existence of exactly one zero of g in the interval, and a sign change of $\gcd(f, g)$ between the boundaries implies the existence of a common zero within the interval. (This zero must then be unique and identical to ϑ.)

For this 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 applies here, too.

2.4.3 Computing with square roots

In a few special cases of the geometric analyses of the next section, we need to compute with square roots $\sqrt{D}$, $D > 0$, although most of the analyses (in particular the generic cases) are performed with integer and rational arithmetic alone. Two approaches exist for computing with square roots:

Separation bound number types like `CORE::Expr` [48] and `leda::real` [18]. These number types support algebraic expressions, among others those built up with field operations and $\sqrt{}$ from integer operands. They implement comparisons by guaranteed numerical approximations combined with separation bounds to determine at which point the precision suffices to detect equality. Comparisons work best for expressions of small nesting depth and large difference in value.

Symbolic representation. The general method of computing modulo the minimal polynomial of an algebraic number (as mentioned in §2.2.4, cf. [60]) reduces in this case to the high-school "pencil and paper" representation $\mathbb{Q}(\sqrt{D}) = \{a + b\sqrt{D} \mid a, b \in \mathbb{Q}\}$ with the obvious implementation of field operations. After checking that D is not a square, one has $a + b\sqrt{D} = a' + b'\sqrt{D} \Leftrightarrow (a, b) = (a', b')$. This representation is preferable for processes like the Euclidean Algorithm in $\mathbb{Q}(\sqrt{D})[x]$, because there is no issue of growing expression trees.

Our implementation of the geometric analyses from Section 4 uses the symbolic representation for algebraic algorithms. `CORE::Expr` or `leda::real` are used to compare root expressions, but it turns out that at those occasions the expressions will always be unequal. Here we just reuse their numerical approximation routines but do not rely on separation bounds. See §4.2.4, case

“$m = 4$”, for an example how the use of both representations is combined.

3 Geometry of Algebraic Curves

This section summarizes those elements of the geometry of algebraic curves that are necessary for our purposes. There are many books on algebraic curves. Mostly, we follow Gibson's excellent introduction [40]. The classic book by Walker [76] is a very useful and well-readable source. Further references are the textbooks by Bix [13] and Cox/Little/O'Shea [25] and also the advanced books by Brieskorn/Knörrer [14] and Fulton [35].

3.1 Basics

Let K be a field. An *affine algebraic curve* (or just *curve*) is a non-constant bivariate polynomial $f \in K[x, y]$, up to multiplication by a non-zero scalar. Often, we do not distinguish a curve f from its *vanishing locus* $\mathrm{V}_K(f) := \{(x, y) \in K^2 \mid f(x, y) = 0\}$. We assume the reader is familiar with the projective plane, projective algebraic curves $F(x, y, z)$, and the correspondence between affine and projective curves by homogenization $F(x, y, z) = z^{\deg(f)} f(x/z,\ y/z)$ and dehomogenization $f(x, y) = F(x, y, 1)$. (If not, see [40].)

Let A be an affine change of coordinates. Recall that $A(\mathrm{V}_K(f)) = \mathrm{V}_K(f \circ A^{-1})$ because $f(v) = 0 \ \Leftrightarrow\ (f \circ A^{-1})(Av) = 0$. All quantities we define below transform in the obvious fashion under coordinate changes.

Let $f \in K[x, y]$ be an algebraic curve, 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 $e_i > 0$ analogous to (3). This corresponds to a decomposition of the zero set f into subsets p_i, each of which we call an *(irreducible) component* of f with *multiplicity* e_i.

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, $\mathrm{mult}(v; f) > 0 \ \Leftrightarrow\ v \in f$. (Notice that f on the left denotes the polynomial and on the right its vanishing locus.) By Taylor expansion around $v = (v_1, v_2)$, the multiplicity is the degree of the lowest-order terms of f expressed in powers of $(x + v_1)$ and $(y + v_2)$. From this characterization, it is immediate that

$$\mathrm{mult}(v; gh) = \mathrm{mult}(v; g) + \mathrm{mult}(v; h); \tag{13}$$

in particular, the multiplicity of a point is at least the sum of the multiplicities of the components containing it. We call the points of multiplicity 1 the *regular* points of f. The points v of multiplicity ≥ 2, i. e., those with $\nabla f(v) = (0, 0)$, are called *singular*.

The linear factors over $\overline{K}$ of the lowest-order terms of f expressed in powers of $(x + v_1)$ and $(y + v_2)$ are called the *tangents* of f at v. A point is called *vertical* if it has a vertical tangent. If v is regular, the unique tangent of f at v has the normal vector $\nabla f(v)$ and hence coincides with the notion of tangent from differential geometry.

Multiplicities of components are irrelevant for the zero set, hence we simplify matters and demand f to be square free from now on.

Proposition 13 *Almost all points v of a square-free y-regular algebraic curve f satisfy $f_y(v) \neq 0$.*

Proof. Given y-regularity, squarefreeness is equivalent to coprimality of f and f_y (cf. §2.2.3). Hence $\mathrm{res}(f, f_y, y) \neq 0$ has only finitely many zeroes ξ, and there are only finitely many points (ξ, η) of f. □

Corollary 14 *Almost all points of a square-free algebraic curve are regular.*

Now we turn to the intersection of two coprime algebraic curves $f, g \in K[x, y]$. Again, a notion of multiplicity will be important. By coprimality, $f \cap g$ is finite. A generic choice of coordinates puts the points of $f \cap g$ into bijective correspondence with the roots of $\mathrm{res}(f, g, y)$. We define the *intersection multiplicity* $\mathrm{mult}(v; f, g)$ 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 $\mathrm{res}(f, g, y)$, and as 0 for $v \notin f \cap g$. This definition is independent of the concrete choice of generic coordinates, but that is not obvious, see [25, 8.7] or [14, 6.1]. This definition extends to projective algebraic curves by dehomogenization. Inspecting the degree of the resultant used in the definition above [40, 14.4] yields:

Theorem 15 (Bézout'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 the projective plane over K, counted with multiplicities.

Corollary 16 *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 $\deg(\mathrm{res}(f, g, y)) \leq \deg(f) \deg(g)$.*

Intersection multiplicity depends on the multiplicity of the point on either curve [40, 14.5]:

Proposition 17 (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 $\operatorname{mult}(v; f, g) \geq \operatorname{mult}(v; f) \operatorname{mult}(v; g)$*, and equality holds iff f and g have no common tangent at v.*

In particular, the tangent of f at a regular point v is the unique line h with $m := \operatorname{mult}(v; f, h) \geq 2$. Generically, one has $m = 2$. In the special case $m > 2$, the regular point v is called a *flex*. Unless f contains a line component, the number of flexes is finite and bounded in terms of $\deg(f)$, see [40, 13.1].

3.2 Arcs of an Algebraic Curve

From now on, consider a y-regular curve f over the field $\mathbb{R}$. Its vanishing locus is a closed subset $f \subseteq \mathbb{R}^2$. Since the set $f \cap f_y$ of *critical points* is finite, $f \setminus f_y$ is an open subset of f. From the Implicit Function Theorem, it follows that every connected component of $f \setminus f_y$ is a parametrized curve

$$\begin{aligned} \gamma_i \colon \left]l_i, r_i\right[&\to \mathbb{R}^2 \\ x &\mapsto (x, \varphi_i(x)) \end{aligned} \tag{14}$$

with some analytic function φ_i (called the *implicit function*) and interval boundaries $l_i, r_i \in \mathbb{R} \cup \{\pm\infty\}$. In particular, every connected component of $f \setminus f_y$ is a C^∞-manifold of dimension 1 which is homeomorphic to an open interval. We call the topological closure $A_i := \operatorname{cl}(\gamma_i(]l_i, r_i[))$ in $\mathbb{R}^2$ of each such component an *arc* of f. Since f is closed, $A_i \subseteq f$.

A generic change of coordinates makes $f_y(v) \neq 0$ for a regular point v, demonstrating that f is a manifold around a critical point v that is not a singular point of f. A singular point v, however, is singular and hence critical in any coordinate system.

The parametrization (14) involves a function φ_i which can be expressed as a convergent power series locally around any x in its domain. Such a power series is a special case of the more general notion of *Puiseux series* (a kind of series involving fractional powers of x) that allow parametrizations even at critical points, see [76, IV.] and [14].

The intersection multiplicity of two curves, defined above in terms of resultants, measures the similarity of these implicit power series.

Proposition 18 *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 an intersection point of f and g that is critical on neither of them. Then* $\operatorname{mult}(v; f, g)$ *is the smallest exponent d for which the coefficients of $(x - v_1)^d$ in the implicit power series of f and g around v*

disagree.

Proof. (Idea only.) Choose a generic coordinate system. Consider f and g as univariate polynomials in y whose zeroes can be written as power series in $x - v_1$. Apply Proposition 7 to see that the multiplicity of v_1 as zero of $\mathrm{res}(f, g, y)$ is the vanishing order of the intersecting pair $(\alpha(x-v_1)-\beta(x-v_1))$ of arcs. See [76, Thm. 5.2] for a generalization and its proof. □

The following corollary is immediate.

Corollary 19 *In the situation of Proposition 18 for $K = \mathbb{R}$, the two arcs of f and g intersecting at a point v change sides iff* $\mathrm{mult}(v; f, g)$ *is odd.*

By changing coordinates, the proposition and its corollary extend to intersections in non-singular critical points.

Now let us inspect the endpoints of an arc $A_i := \mathrm{cl}(\gamma_i(]l_i, r_i[))$ more closely. The y-regularity of f implies that f has no vertical asymptotes. If $r_i < +\infty$, then A_i has a *right endpoint* $(r_i, \lim_{x \to r_i} \varphi_i(x)) \in \mathbb{R}^2$ which is a critical point of f. If $r_i = +\infty$ then A_i has a *right endpoint at infinity.* We use analogous definitions for the *left endpoint.*

Criticality of a point (r, s) can be characterized in terms of substituting a vertical line in parametric form $t \mapsto (r, s + t)$ info f. Taylor's formula yields:

$$f(r + 0, s + t) = f(r, s) + f_y(r, s)t + \frac{1}{2}f_{yy}(r, s)t^2 + \dots \qquad (15)$$

We obtain:

Lemma 20 *Let $f \in K[x, y]$ be a y-regular algebraic curve, and let $(r, s) \in K^2$. 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 $y = s$.

It follows immediately that no two critical points on a curve of degree ≤ 3 can have the same x-coordinate.

Let us now distinguish several kinds of critical points. A critical point (r, s) of f is either a singular or a regular point of f. In the latter case, the tangent h to f at (r, s) is the vertical line $h = x - r$. By reasoning analogous to Proposition 18, we see that $\mathrm{mult}((r, s); f, h) = \mathrm{mult}(0; f(r, s + t))$. From (15) we obtain that (r, s) is a flex iff $f_{yy}(r, s) = 0$. If (r, s) is not a flex, it follows from Corollary 19 that f is locally on one side of the vertical tangent at (r, s),

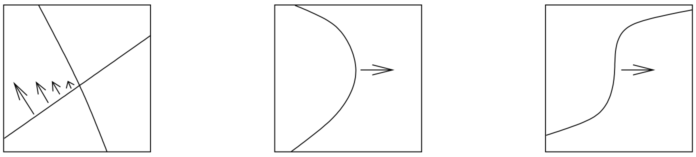

Fig. 2. The three kinds of critical points: singularity (left), x-extremality (middle), and vertical flex (right). The arrows indicate the gradient.

meaning that the point (r, s) has a locally minimal or maximal x-coordinate on f, and we call it a *left* or *right*, resp., *x-extreme point*.

In the case of cubic curves, the tangent at (r, s) can intersect f with multiplicity at most 3, so that in case of (r, s) being a flex the curve must intersect with multiplicity exactly 3 and change sides with its tangent.

In summary, we arrive at distinguishing the following kinds of critical points:

(1) **Singularity:** The point satisfies $f_y = f_x = 0$, and an arbitrary number of arcs (maybe zero) meet there.
(2) **x-Extremality:** The point satisfies $f_y = 0$ and $f_x, f_{yy} \neq 0$. Two arcs meet there, both of which lie on the same side of the vertical tangent.
(3) **Vertical Flex:** The point satisfies $f_y = f_{yy} = 0$, $f_x \neq 0$, and joins two arcs. For cubics, the two arcs are known to lie on different sides of the vertical tangent.

If the union of two arcs incident to a point $v \in f$ form a C^∞-manifold around v, we call this pair of arcs a *branch* of f at v. A curve has exactly one branch at any regular point (including those that are critical) and can have an arbitrary number of branches at a singularity.

If a point $v \in f$ is x-extreme, then $\operatorname{mult}(v; f, f_y) = 1$ by Proposition 17, because $f_{yy}(v) \neq 0$ is equivalent to f_y being regular at v with a non-vertical tangent. The converse argument shows that $\operatorname{mult}(v; f, f_y) \geq 2$ if v is a singularity or a vertical flex. For cubics, we can be more precise:

Proposition 21 *Let $f \in \mathbb{C}[x, y]$ be a y-regular cubic curve with a vertical flex $(r, s) \in \mathbb{C}^2$. The multiplicity of r as a zero of* $\operatorname{res}(f, f_y, y)$ *is exactly 2.*

Proof. After a suitable translation, $(r, s) = (0, 0)$. Then f can be written as $f(x, y) = y^3 + a_1xy^2 + b_2x^2y + c_3x^3 + b_1xy + c_2x^2 + c_1x$ with $c_1 \neq 0$. One obtains $\operatorname{res}(f, f_y, y) = 27c_1^2x^2 +$ higher-order terms. □

3.3 Classification of Cubic Curves

In the following paragraphs, we classify square-free real algebraic curves $f \in \mathbb{Q}[x, y]$ of degree at most 3 with respect to their decomposition into components as well as number and kind of their singularities. The statements below follow from three sources:

- The textbook classification of conics and cubics [40] up to complex-projective changes of coordinates.
- Conjugacy arguments to demonstrate that something is real or rational or algebraic of at most a certain degree. For example, let the homogenization F of f have exactly three singularities in the complex-projective plane. They are the solutions of $F_x = F_y = F_z = 0$. Complex solutions come in pairs of conjugates, so one of the three solutions must be real. Algebraic solutions of degree d come in groups of d conjugates (as discussed in §2.2.4), so the number of solutions, in this case 3, bounds their algebraic degree.
- Real-projective changes of coordinates to bring cubic curves with a singularity into one of three normal forms [13, Thm. 8.4] that expose the real geometry of the singularities.

3.3.1 Lines and Conics

Let $f \in \mathbb{Q}[x, y]$ be a line, i. e., $\deg(f) = 1$. It is irreducible, real, and it possesses no singularities.

Let $f \in \mathbb{Q}[x, y]$ be a conic. It is either irreducible or a pair of two distinct lines. An irreducible conic possesses no singularities and is either the empty set (like $x^2 + y^2 + 1$) or one of ellipse, parabola, or hyperbola.

By (13) in conjunction with Bézout's Theorem, a line pair $f = g_1 g_2$ has a unique singularity in the projective plane, i. e., the unique intersection point v of its two components g_1 and g_2, which is rational but may lie at infinity. The two tangents at v are g_1 and g_2. If they are real, v is a *crunode*, and two real branches intersect at v. If they are complex, v is an *acnode*, which is an isolated point of f in the real plane.

3.3.2 Irreducible Cubics

Let $f \in \mathbb{Q}[x, y]$ be an irreducible cubic. It has at most one singular point v in the projective plane. If v exists, f is called *singular*, otherwise *non-singular*.

The unique singularity v of a singular irreducible cubic f is rational but may lie at infinity. The curve f has exactly $\operatorname{mult}(v; f) = 2$ tangents at v. If the

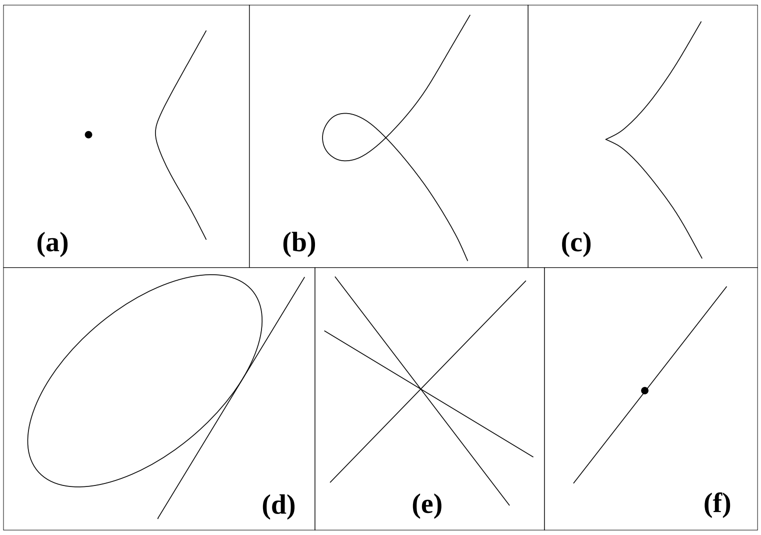

Fig. 3. Kinds of singularities: **(a)** acnode, **(b)** crunode, **(c)** cusp, **(d)** tacnode, **(e)** real triple point, **(f)** complex triple point

tangents are distinct, v is a *crunode* (two real branches and two real tangents) or an *acnode* (two complex-conjugate branches and tangents). If there is one double tangent, it is real, and v is called a *cusp*. At a cusp, two arcs converge from one side and do not continue in the real plane. (They come out complex-conjugate on the other side.)

3.3.3 Cubics with several components

Let $f \in \mathbb{Q}[x, y]$ be a cubic that is not irreducible. Then it has two components (a line and an irreducible conic) or three components (three lines). Since lines and irreducible conics have no singularities, by (13) 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 Bézout's Theorem, g and h intersect in exactly two complex-projective points without common tangent or in exactly one point with a common tangent. In the former case, the two intersections may be complex-conjugates, so that there are no singularities in the real plane, or two real *crunodes*. In the latter case, g is a tangent to h at the unique intersection point, and the resulting intersection is a *tacnode*. By uniqueness, a tacnode is rational. The two crunodes are rational, or they are algebraic of degree 2 and conjugates of each other. Both components g and h have rational coefficients, because $g = y + ax + b$ is the unique solution to $\mathrm{res}(y + ax + b, f, y) = 0$ and thus rational, and then the same holds for $h = f/g$.

Now assume $f = g_1g_2g_3$ consists of three distinct lines. Exactly one or three of them are real; and in both cases, two or three of them might have algebraically conjugate equations.

If these three lines are concurrent at one point $v \in g_1 \cap g_2 \cap g_3$ (maybe at infinity), then v is the unique (and hence rational) singularity of f with $\mathrm{mult}(v; f) = 3$ and with g_1, g_2, and g_3 as its three distinct tangents. In this case, we call f a *star*. Depending on all tangents being real or not, v is called a *real* or *complex triple point*.

If the lines are non-concurrent, they form a *real* or *complex triangle*, i. e., for all choices $\{i, j, k\} = \{1, 2, 3\}$ of three distinct indices there is exactly one intersection point $s_k \in g_i \cap g_j$, $s_k \notin g_k$. The point s_k is real (rational) iff g_k is real (or rational, resp.). A real triangle has three real singularities, at most one of which may lie at infinity. Its singularities are *crunodes* whose coordinates are algebraic of degree 1, 2, or 3. A complex triangle has exactly one singularity in the real affine plane which is an *acnode*. Its coordinates are algebraic of degree 1 or 3.

4 Geometric Analyses

This section describes methods to analyze the geometry of a real algebraic curve $f \in \mathbb{Q}[x, y]$ of degree $\deg(f) \leq 3$, and of pairs of such curves. We can restrict ourselves to integer coefficients for the sake of efficiency without limitation of generality. The result of the analysis resembles a cylindrical algebraic decomposition of the plane (cf. Arnon et al. [5,6] and the textbook [8, 12.5]) with additional information on adjacencies and intersection multiplicities, but our method to obtain it is purpose-built and optimized for our application. It avoids arithmetic with algebraic numbers of high degree by using geometric properties of the curves themselves and, if necessary, auxiliary curves.

4.1 Analysis of a Cubic Curve

The general view we take on the geometry of f 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 other words: How does the intersection of f with a vertical line $g = x - \xi$ evolve as we sweep g from $-\infty$ to $+\infty$?

The answer was given abstractly in the previous section: Over almost all x-coordinates, the arcs evolve smoothly according to their implicit functions and do not change their number and relative position. Only at the x-coordinates of critical points $v \in f \cap f_y$, something happens. Hence we call these points *(one-*

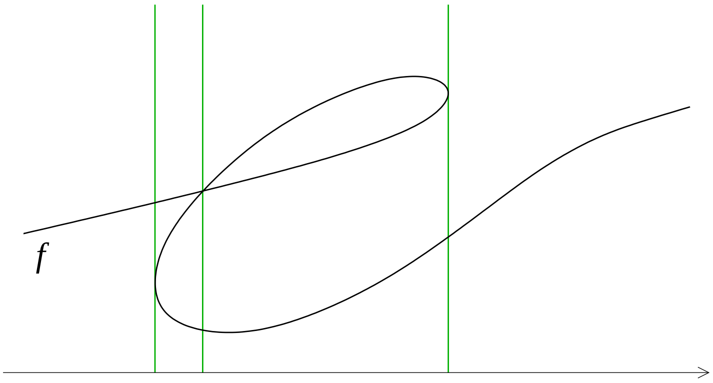

Fig. 4. The event x-coordinates of a curve induce a partition of the plane into vertical lines over event x-coordinates and vertical stripes over the intervals between them.

curve) event points and their x-coordinates *(one-curve) event x-coordinates.*

Below, we describe an algorithm that accepts an algebraic curve $f \in \mathbb{Z}[x, y]$ of degree $\deg(f) \leq 3$ 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.
- 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 degrees 2 and 3.

Locating the event points amounts to intersecting f and f_y. Our general approach is to compute the event x-coordinates with a resultant, but to avoid the costly arithmetic involved in computing the symbolic representation of the matching y-coordinates, where possible. Instead, we stick to a "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 the last author's thesis [78] and its references. In the same fashion, we do not care about the actual parametrizations of arcs, just their relative position. Computing numerical coordinate data can take place as a post-processing step (cf. §4.4.3).

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

- The curve f is y-regular.
- The curve f is square free.
- No two points of $\mathrm{V}_{\mathbb{C}}(f) \cap \mathrm{V}_{\mathbb{C}}(f_y)$ 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 is called *vertical* 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 Lemma 20.) Violations are signalled to the caller and cause the algorithm to abort. It is then the caller's responsibility to establish the conditions and to restart the algorithm. The violation of a condition does not cause an incorrect result.

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. 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 genericity conditions on the coordinate system, see §4.4. Since the conditions can be established mechanically, the resulting algorithm remains complete. Choosing generic coordinates is a standard trick. It is ubiquitous in proofs (cf. any of [76], [40] and [14]) and often used for algorithms (see, e.g., [70] or [42]).

4.1.1 Event x-Coordinates

If f is not y-regular, signal "not y-regular" and abort.
Compute $R_f := \mathrm{res}(f, f_y, y)$. If $R_f = 0$, signal "not square free" and abort. Compute a square-free factorization $R_f = \prod_{m=1}^{M} R_{fm}^m$. Isolate the real zeroes of each non-constant factor and sort them to obtain the ordered sequence $x_1 < x_2 < \ldots < x_n$ of one-curve event x-coordinates. They correspond bijectively to the one-curve event points $f \cap f_y$. Record the multiplicity m_i of each x_i.

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

$$\mathbb{R} \;=\;]-\infty, x_1[\cup \{x_1\} \cup]x_1, x_2[\cup \{x_2\} \;\cup\; \ldots \;\cup\; \{x_n\} \cup]x_n, +\infty[. \quad (16)$$

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.

Compute a rational sample point $r_i \in]x_{i-1}, x_i[\cap \mathbb{Q}$ within each interval between events. Count the real zeroes of $f(r_i, y) \in \mathbb{Q}[y]$. This determines the number k_i of arcs of f over the i-th interval between events. Since they are known to be disjoint, this gives a complete description of the behaviour of f over the interval. With respect to a specific interval between events, we identify the arcs over it by *arc numbers* from 1 to k_i, counted in ascending order of y-coordinates.

In case $\deg(f) = 3$, it remains to check the absence of vertical flexes $v \in$

$f \cap f_y \cap f_{yy} \setminus f_x$. If $R := \text{res}(f_y, f_{yy}, y) \neq 0$, this amounts to computing with the intersection points of a line and a conic, involving coordinates of algebraic degree ≤ 2, and checking that f_x vanishes at these points if f does. If $R = 0$, then $f_y = \frac{1}{2} f_{yy}^2$ is a double line; and we know there is no vertical flex iff the number of real roots of R_{f2} (cf. Proposition 21) is equal to the number of real roots of $\gcd(R_{f2}, \text{res}(f_x, f_{yy}, y))$. If a vertical flex exists, signal "vertical flex" and abort.

4.1.2 Arcs over event points

The rest of this section is concerned with the analysis of f at event points (x_i, y_i). We say an arc is *involved* in an event if the event point is contained in the arc. (If this holds, the event is one of the arc's endpoints.) Otherwise, the arc is called *uninvolved* or *continuing.*

For each event x-coordinate x_i, we will determine:

- The number k'_i of distinct points on f over x_i.
- The kind of event (left/right x-extreme or kind of singularity).
- 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 (if any).

Arc numbers over x_i are defined by counting the points of f over x_i in ascending order of y-coordinates without multiplicities and thus range from 1 to k'_i. For $\deg(f) = 2$, we have $k'_i = 1$, and all arcs over incident intervals are involved. For $\deg(f) = 3$, we have either $k'_i = 1$, in which case all arcs over incident intervals are involved, or we have $k'_i = 2$, so that there is exactly one continuing arc, and we have to find out whether the non-event point (x_i, y'_i) on it lies above or below the event point (x_i, y_i).

The analysis of an event is split into three parts. If $m_i = 1$, use the method of §4.1.3. For $m_i > 1$, use the methods of §4.1.4 or §4.1.5, depending on $s := \sum_{m \geq 2} \deg(R_{fm})$, i. e., the number of singular points of f in $\mathbb{C}^2$.

4.1.3 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 first distinction is made easily by the sign of $k_{i+1} - k_i = \pm 2$. Let us assume the sign is positive. Then we have a left x-extreme point. The opposite case is symmetric. For $\deg(f) = 2$, this completes the analysis.

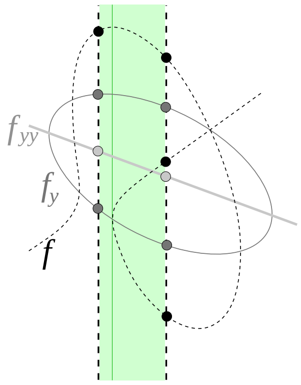

Fig. 5. The situation at interval boundaries $[r_-, r_+]$ close to a left x-extreme point.

The second distinction amounts to deciding whether the uninvolved arc in case $\deg(f) = 3$ lies above or below the x-extreme point. We use f_y and f_{yy} as auxiliary curves to make this decision. Over x_i, there are two arcs of f_y: one containing (x_i, y_i), because it is a double root of $f(x_i, y)$, another in between (x_i, y_i) and the continuing arc of f by the Mean Value Theorem. This implies $\operatorname{res}(f_y, f_{yy}, y)(x_i) \neq 0$. Hence we can refine the interval $]x_i, x_{i+1}[$ to an interval $]r_-, r_+[\ni r_i$ with rational endpoints such that $[r_-, r_+]$ does not contain a root of $\operatorname{res}(f_y, f_{yy}, y)$.

At x_i and thus over the whole interval $[r_-, r_+]$, both arcs of f_y lie on the same side of the continuing arc of f. Let us compute over r_- which side it is. Between the two zeroes of $f_y(r_-, y)$ lies the unique zero c of the linear polynomial $f_{yy}(r_-, y) \in \mathbb{Q}[y]$. Its relative position to the unique zero of $f(r_-, y)$ is determined by the sign of $f(r_-, c)$: The point (r_-, c) and therefore also the x-extreme point (x_i, y_i) lies above/below the continuing arc of f iff $f(r_-, c)$ agrees/disagrees in sign with the leading coefficient of f.

4.1.4 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 read off $x_i \in \mathbb{Q}$ immediately from the linear resultant factor R_{fm_i}. Next, $y_i \in \mathbb{Q}$ can be obtained by factoring $f(x_i, y) \in \mathbb{Q}[y]$ by multiplicities. Let $\hat{f}(x, y) = f(x_i + x, y_i + y)$. Group the terms in $\hat{f} = \hat{f}_3 + \hat{f}_2$ by their degrees. Constant and linear part vanish since $\operatorname{mult}((0, 0); \hat{f}) \geq 2$. Either $\hat{f}_3$ or $\hat{f}_2$ might also be zero. Two observations allow us to take shortcuts in computing $\hat{f}$: The highest-order terms of f are invariant under translation. For $d = 3$, the quadratic part $\hat{f}_2$ can be computed by evaluating partial derivatives according to Taylor's formula.

For a conic $\hat{f} = ay^2 + bxy + cx^2$, the kind of singularity is all that needs to be determined. Let $\sigma = \operatorname{sign}(b^2 - 4ac)$. 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.

For a cubic, compute $\hat{f}(x, y) = f_3(x, y) + ay^2 + bxy + cx^2$. If $a = b = c = 0$, we have a triple point: It is a real triple for $k_i = 3$ and a complex triple for $k_i = 1$, and it involves all $k_i = k_{i+1}$ arcs on both sides.

Otherwise, let $\sigma = \operatorname{sign}(b^2 - 4ac)$ and distinguish these cases:
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.

The sign of a discriminant of a quadratic (or cubic) equation can distinguish the cases of 0 versus 2 (or 1 versus 3) simple real roots [8, Cor. 4.19]. At various places in the method above one can trade computing arc counts k_i against determining the sign of a discriminant. One could even exploit that for all $\xi \in \mathbb{R}$, $R_f(\xi)$ is the discriminant of $f(\xi, y) \in \mathbb{R}[y]$ up to a constant factor and iteratively compute k_{i+1} from k_i and $m_i \bmod 2$. In terms of running time for arrangement computation, all those choices do not make a difference, because curve pair analyses dominate the analyses of individual curves.

4.1.5 Multiple Singularities

Assume the curve f has exactly $s > 1$ singular points in $\mathbb{C}^2$. Let exactly r of them be real. 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.

From §3.3, we know that $\deg(f) = 3$ and that (x_i, y_i) is an acnode or crunode resulting from the intersection of two components. By Lemma 20 we know $f(x_i, y) = \ell(f)(y - y_i')(y - y_i)^2$ for some $y_i' \in \mathbb{R}$. The required nonverticality of (x_i, y_i) is equivalent to $y_i \neq y_i'$, because $y_i = y_i'$ implies a threefold intersection and hence tangency of the vertical line $x - x_i$ by Proposition 17. Thus 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 indeed?
- 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 $\mathrm{sign}(y_i - y_i')$. We will first construct a polynomial $\delta(x) \in \mathbb{Q}[x]$ such that $\delta(x_i) = y_i - y_i'$. Then we discuss its sign at x_i.

Observe that the x-coordinates of all singularities of f in $\mathbb{C}^2$ are precisely the zeroes of the square-free polynomial $h = R_{f2}R_{f3}R_{f4}$. Let ϑ be any of them. It is well-known how to do arithmetic in the extension field $\mathbb{Q}(\vartheta)$ by computing in $\mathbb{Q}[x]$ modulo h (see §2.2.4, cf. [60]). With some care, this is possible even if h is not irreducible. Our idea is to perform the Euclidean Algorithm for $f|_\vartheta$ and its derivative $f_y|_\vartheta$ modulo h to obtain linear factors for the double and the simple root of $f|_\vartheta$. Dividing by f_y w. r. t. y is easy since its leading coefficient is a constant. We expect the remainder to be the linear factor $\alpha y + \beta$ belonging to the desired double root $-\beta/\alpha$. If no choice of ϑ is a root of α, we can compute a representative for $1/\alpha$ modulo h by an extended gcd computation. If, on the other hand, one choice of ϑ makes α vanish, then it makes β vanish as well (because $\deg(\gcd(f|_\vartheta, f_y|_\vartheta)) = 0$ is impossible), and $f|_\vartheta$ has a triple root. This means ϑ is the x-coordinate of a complex vertical singularity. By y-regularity, this cannot happen for a triangle, hence f is of type "conic and line" and the offending singularity is real, violating the nonverticality condition.

Compute $\delta(x)$ as follows:
Do polynomial division $f = qf_y + g$ w. r. t. y to obtain $g = \alpha y + \beta$.
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.
Factor $f|_\vartheta = \ell(f)\varphi_1\varphi_2^2$ by setting
$\varphi_2(y) := ug \bmod h = y + \eta_2$ and $\varphi_1(y) := f/(\ell(f)\varphi_2^2) \bmod h = y + \eta_1$.
Let $\delta(x) = \eta_1(x) - \eta_2(x) \in \mathbb{Q}[x]$.

With δ at hand, let us return to the problem of analyzing the singularity (x_i, y_i). If $s = \deg(h) = 2$, then both singularities are real, we solve h for x_i, and evaluate the non-zero sign of $\delta(x_i)$ straight away. Since there is a real line component joining the two real singularities, (x_i, y_i) is a crunode.

If $s = 3$, then f is a triangle with no vertex at infinity. Let us first consider the case that all vertices are real and have x-coordinates $x_1 < x_2 < x_3$. (All of them are crunodes.) It is obvious from the shape of a triangle that $\mathrm{sign}(\delta(x_1)) \neq \mathrm{sign}(\delta(x_2)) \neq \mathrm{sign}(\delta(x_3))$. But we have $\deg(\delta) \leq \deg(h) - 1 = 2$, so that $\mathrm{sign}(\delta(x_1)) = \sigma$, $\mathrm{sign}(\delta(x_2)) = -\sigma$, and $\mathrm{sign}(\delta(x_3)) = \sigma$ for $\sigma := \mathrm{sign}(\ell(\delta))$. Hence σ contains the complete answer if all vertices are real: The continuing arc runs above the singularity iff $(-1)^i\ell(\delta) > 0$.

In fact, the same holds if just one vertex (x_1, y_1) is real. (That vertex is an acnode.) For brevity, we just sketch the proof: By translating (x_1, y_1) to $(0, 0)$, scaling coordinates with appropriate positive factors, and making $f = gh$ monic, one obtains a real line $g(x, y) = y + ax + c$ with $c = g(0, 0) \neq 0$ and

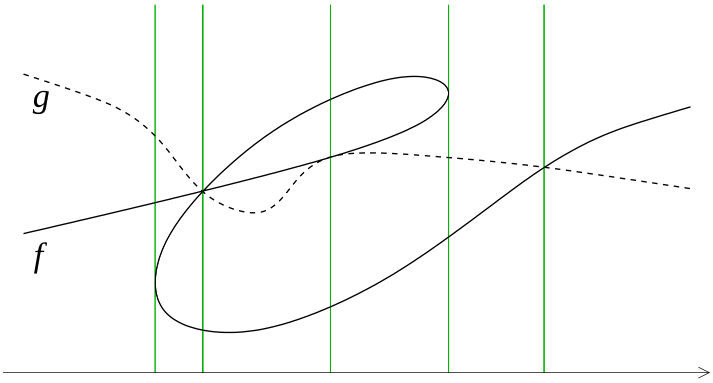

Fig. 6. The event x-coordinates of a curve pair induce a partition of the plane into vertical lines over event x-coordinates and vertical stripes over the intervals between them.

a complex line pair $h(x, y) = y^2 + bxy + x^2$ with discriminant $b^2 - 4 < 0$. Following the construction of δ, one obtains

$$\delta(x) = \frac{3}{c}(a^2 - ba + 1)x^2 + (4a - 2b)x + c. \tag{17}$$

The factor $\lambda(a) = a^2 - ba + 1$ of $\ell(\delta)$ has discriminant $b^2 - 4 < 0$ and hence is 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.

4.2 Analysis of a Pair of Cubic Curves

Let us turn to the geometric analysis of a pair $\{f, g\} \subseteq \mathbb{Z}[x, y]$ 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 *two-curve event points* at which this changes are the one-curve event points on either curve and the intersection points, because $f \cup g$ has the equation fg and the critical points

$$fg \cap (fg)_y = (f \cap f_y) \cup (g \cap g_y) \cup (f \cap g). \tag{18}$$

(This equality is easily derived from $(fg)_y = f_y g + f g_y$.)

We give an algorithm that takes a pair $\{f, g\}$ of algebraic curves $f, g \in \mathbb{Z}[x, y]$ 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.
- f and g are coprime.
- No two points of $\mathrm{V}_{\mathbb{C}}(f) \cap \mathrm{V}_{\mathbb{C}}(g)$ 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 $\{f, g\}$ (see §4.2.4) is y-regular. For any $v \in \mathbb{R}^2$ with $\mathrm{mult}(v; f, g) = 2$ it holds that: The complex intersections $\mathrm{V}_{\mathbb{C}}(J) \cap \mathrm{V}_{\mathbb{C}}(f)$ and $\mathrm{V}_{\mathbb{C}}(J) \cap \mathrm{V}_{\mathbb{C}}(g)$ do not contain a point covertical to v, and the set of complex one-curve events $\mathrm{V}_{\mathbb{C}}(J) \cap \mathrm{V}_{\mathbb{C}}((J/\gcd(J, J_y))_y)$ does not contain a point covertical or equal to v.
- If f is a complex triangle whose acnode v is a regular point of g, then f_y and g are coprime and $\mathrm{mult}(v; f_y, g)$ is odd.

The algorithm checks these conditions and signals violations. For our goal of arrangement computation, coprimality is not a restriction, since one can replace two curves f, g in the input to arrangement computation by $h := \gcd(f, g)$, f/h, and g/h if $h \neq 1$. See §4.4 on how to establish the other conditions.

4.2.1 Event x-Coordinates

Analogous to the analysis of one curve, the sorted sequence of two-curve event x-coordinates induces a partition (16) of the x-axis, and again we use terms like "interval between events" and so on. The analysis of a curve pair begins as follows.

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.

Determine a rational sample point $r_i \in]x_{i-1}, x_i[\cap \mathbb{Q}$ for each interval between

events.
Find and sort the real zeroes of $f(r_i, y), g(r_i, y) \in \mathbb{Q}[y]$ to determine the sorted sequence of f-arcs and g-arcs over each interval between events.

Then the event points are analyzed, as detailed in the rest of this section.

4.2.2 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) on one curve, let us say f. We know $(x_i, y_i) \notin g$, and we know g has no one-curve event over x_i.

If (x_i, y_i) is not an isolated point of f, then its position relative to the arcs of f and g can be read off directly from the position of the arcs of f containing (x_i, y_i) over an incident interval between events.

Now assume (x_i, y_i) is an isolated point of f. Such an acnode can occur on a line pair, on a singular irreducible cubic, or on a triangle. In the latter case, it can be algebraic of degree up to 3, discouraging arithmetic with x_i. However, we can use f_y as an auxiliary curve that contains (x_i, y_i) and does not have a one-curve event at x_i. If $\deg(f_y) = 2$, the Mean Value Theorem tells us that f_y has two arcs over x_i, a relevant one containing (x_i, y_i) and another one between (x_i, y_i) and the continuing arc of f. We simply inspect the relative position of the relevant arc of f_y and the arcs of g over a "nearby" rational x-coordinate. "Nearby" means closer than any two-curve event of $\{f_y, g\}$, i. e., not separated from x_i by a zero of $\operatorname{res}(f_y, f_{yy}, y)$, $\operatorname{res}(g, g_y, y)$ or $\operatorname{res}(f_y, g, y)$. If the latter resultant is zero, f_y has a common component $h := \gcd(f_y, g)$ with g. But h cannot contain (x_i, y_i), hence we can just replace f_y by f_y/h and repeat.

4.2.3 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. Hence if $m_i^{(f)} = 1$ or $m_i^{(g)} = 1$, we signal "x-extreme over intersection x-coordinate" and abort. So from now on, 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 §4.2.7.

In the remaining cases, at least one of the polynomials $f|_{x_i}, g|_{x_i} \in \mathbb{R}[y]$ is square free by Lemma 20 so that $d := \deg(\gcd(f|_{x_i}, g|_{x_i}))$ is the number of their distinct common complex zeroes. Hence the noncoverticality condition for intersection points holds iff $d = 1$. By Proposition 8 and y-regularity of f and g, this is equivalent to $\mathrm{sres}_1(f, g, y)(x_i) \neq 0$. We can check this condition without explicit arithmetic in x_i by verifying that the gcd of the subresultant and the resultant factor defining x_i has equal signs at r_i and r_{i+1}. If not, we signal "covertical intersections" and abort. 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 – §4.2.4.
- Exactly one of the curves has a singularity over x_i for which a rational representation is known – §4.2.5.
- Exactly one of the curves has a singularity over x_i for which a rational representation is not known – §4.2.6.
- Both curves have a singularity over x_i – §4.2.7.

The descriptions of the two asymmetric cases assume w. l. o. g. that the singularity is on f.

4.2.4 Intersection regular-regular

Let x_i be a two-curve event x-coordinate with $m := m_i^{(fg)} > 0$ and $m_i^{(f)} = m_i^{(g)} = 0$. If $m = 1$ or after checking $\mathrm{sres}_1(f, g, y)(x_i) \neq 0$ (cf. §4.2.3), we know there is exactly one intersection point of f and g.

By Corollary 19, 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 (see Figure 7(a)).

If m is even, the arc sequences over the incident intervals are equal. However, since $\deg(R_{fg}) \leq 9$ by Corollary 16, 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 $\mathbb{Q}(\sqrt{D})$, $D > 0$: Compute $h := \gcd(f|_{x_i}, g|_{x_i}) = y - y_i$, using symbolic arithmetic with $\sqrt{D}$ if necessary. Then sort the roots of $f|_{x_i}/h$, $g|_{x_i}/h$, and h. They are pairwise distinct and can be expressed as rationals or as one-root numbers using `leda::real` or `CORE::Expr`.

The remaining case $m = 2$ can be tackled using the Jacobi curve [78,79]. 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 $J := \det(\nabla f,\ \nabla g) = f_x g_y - f_y g_x$. 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 17.

Later, we need the following technical result that makes the Jacobi curve well-defined without reference to a specific choice of coordinates.

Proposition 22 *Let $f, g \in \mathbb{C}[x, y]$ be two algebraic curves and J their Jacobi curve. Let M be a linear change of coordinates. The Jacobi curve of $f \circ M$ and $g \circ M$ is equal to $J \circ M$, up to multiplication by the non-zero constant* $\det(M)$.

Proof. $\nabla(f \circ M) = M^{\mathsf{T}}((\nabla f) \circ M)$ and $\det(M^{\mathsf{T}} N) = \det(M) \det(N)$. □

Now we state the main theorem of [79] specialized to our case $m = 2$ and give a more geometric proof:

Theorem 23 *Let $f, g \in \mathbb{Q}[x, y]$ be two real algebraic curves, and let v be a non-critical point of both f and g such that* $\mathrm{mult}(v; f, g) = 2$. *Then their Jacobi curve $J = f_x g_y - f_y g_x$ is indeed a curve (i. e., non-constant) and has intersection multiplicities* $\mathrm{mult}(v; f, J) = \mathrm{mult}(v; g, J) = 1$.

Proof. Let $y = \alpha(x)$ and $y = \beta(x)$ be the implicit functions of f and g, resp., around v. Differentiating both sides of $0 = f(x, \alpha(x))$ twice w. r. t. x, one obtains

$$\begin{aligned} \alpha'(x) &= -\frac{f_x}{f_y}(x, \alpha(x)) \\ \alpha''(x) &= -\frac{1}{f_y^3}\Big(f_{xx} f_y^2 - 2 f_{xy} f_x f_y + f_{yy} f_x^2\Big)(x, \alpha(x)). \end{aligned}$$

Choose w. l. o. g. a scalar multiple of f such that $f_y < 0$ around v. Then the *curvature* of f at $(x, \alpha(x))$ is

$$\begin{aligned} \kappa_f(x, \alpha(x)) := \frac{\alpha''(x)}{(1 + \alpha'(x)^2)^{3/2}} &= \frac{f_{xx} f_y^2 - 2 f_{xy} f_x f_y + f_{yy} f_x^2}{(f_x^2 + f_y^2)^{3/2}}(x, \alpha(x)) \\ &= \Big(\frac{1}{\|\nabla f\|_2^3} (\nabla f^{\perp})^{\mathsf{T}} \mathrm{H}_f (\nabla f^{\perp})\Big)(x, \alpha(x)), \end{aligned} \tag{19}$$

where

$$\mathrm{H}_f = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}$$

is the Hessian matrix of f, and $(a, b)^{\perp} = (-b, a)$ denotes orthogonal complement. By Proposition 18, $\mathrm{mult}(v; f, g) = 2$ implies $\alpha'(v_1) = \beta'(v_1)$ and

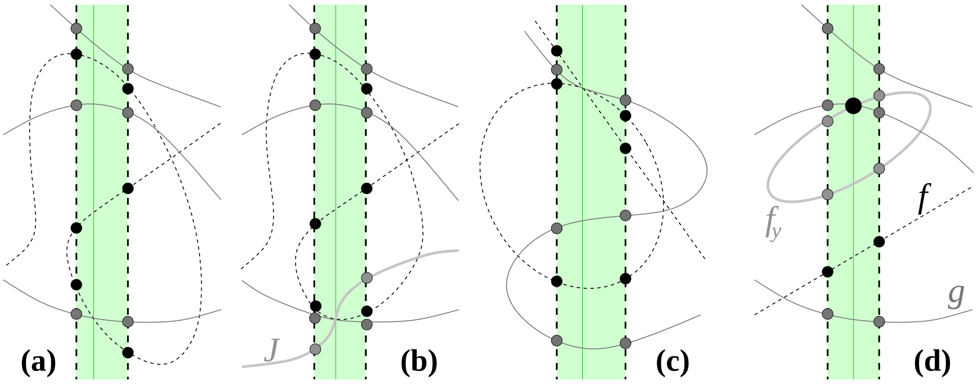

Fig. 7. Determining the arcs involved in an intersection v of f and g: **(a)** If v is regular and m is odd, one can observe a transposition. **(b)** If v is regular and $m = 2$, we use the Jacobi curve J. **(c)** If v is a crunode of f and regular on g, one can observe a transposition. **(d)** If v is an acnode of f and regular on g, we use f_y.

$\alpha''(v_1) \neq \beta''(v_1)$. Hence there exists $\lambda \neq 0$ such that $\nabla f(v) = \lambda \cdot \nabla g(v)$, demonstrating $v \in J$; and furthermore, we have $\kappa_f(v) \neq \kappa_g(v)$.

To prove $J \neq \text{const}$ and $\text{mult}(v; f, J) < 2$, it suffices to show that the Jacobi curve of f and J does not vanish at v. It can be written as

$$\begin{aligned}\begin{vmatrix} f_x & J_x \\ f_y & J_y \end{vmatrix} &= \begin{vmatrix} f_x & f_x g_{xy} - f_y g_{xx} \\ f_y & f_x g_{yy} - f_y g_{xy} \end{vmatrix} - \begin{vmatrix} f_x & g_x f_{xy} - g_y f_{xx} \\ f_y & g_x f_{yy} - g_y f_{xy} \end{vmatrix} \\ &= (\nabla f^{\perp})^{\mathsf{T}} \mathrm{H}_g \nabla f^{\perp} - (\nabla f^{\perp})^{\mathsf{T}} \mathrm{H}_f \nabla g^{\perp}\end{aligned}$$

By (19) and $\nabla f(v) = \lambda \cdot \nabla g(v)$, evaluating at v yields

$$\begin{vmatrix} f_x(v) & J_x(v) \\ f_y(v) & J_y(v) \end{vmatrix} = \lambda^2 \cdot \|\nabla g(v)\|_2^3 \cdot (\kappa_g(v) - \kappa_f(v)) \neq 0,$$

as desired. □

The theorem 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 touching arcs of f and g (see Figure 7(b)), 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 $\{J, f\}$ and $\{J, g\}$ over I except for the intersection 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 components, but these cannot contain (x_i, y_i) and can thus be deleted.

Our algorithm is this:
Compute $J := f_x g_y - f_y g_x$. If J is not y-regular, signal this and abort.

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

If x_i is a zero of R_J or a multiple zero of R_{Jf} or R_{Jg},
signal "forbidden 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 $f(r_+, y), g(r_+, y), J(r_+, y)$. Compare them to detect the pair of an f-arc and a g-arc that both change sides with the same J-arc.

4.2.5 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 $\mathbb{C}^2$.

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 $y_i' \in \mathbb{Q}$ of a continuing arc of f. The noncoverticality condition holds iff $g(x_i, y_i') \neq 0$ or y_i' does not exist. If it is violated, we signal this and abort.

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.

4.2.6 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. Assume x_i is 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, according to the classification from Section 3.3.

After checking $m_i^{(fg)} = 2$ or $\operatorname{sres}_1(f, g, y)(x_i) \neq 0$ (cf. §4.2.3), 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 (x_i, y_i) of f, two branches of f with different tangents intersect. An arc of g passing through (x_i, y_i) can share a tangent with at most one of the branches of f. Thus it changes sides with at least one of them, and we know they change sides with each other. Therefore, a g-arc A intersects f in (x_i, y_i) iff one of the following conditions holds:

- On one side, A lies between the two arcs of f containing (x_i, y_i).
- On one side, A lies below and on the other side, A lies above the respective two arcs of f containing (x_i, y_i).

Hence the analysis of a crunode (x_i, y_i) reduces to inspecting the arc sequences over incident intervals, see Figure 7(c) on page 74.

Let us now consider the case of an acnode (x_i, y_i). It occurs only for a complex triangle. The acnode has no supporting arcs on f, but we can use f_y instead, similar to what we did in §4.2.2. (We described there which arc of f_y contains (x_i, y_i).) The situation is depicted in Figure 7(d). The conic f_y contains the three non-collinear points $f \cap f_y$ but none of their connecting lines, so it is irreducible. This entails $\mathrm{res}(f_y, f_{yy}, y) \neq 0$.

We must check that the intersection of f and g over x_i indeed occurs in the acnode (x_i, y_i) (noncoverticality), and we check that f_y and g are coprime and intersect at (x_i, y_i) with odd multiplicity. The latter is satisfied iff we can observe a transposition of the relevant arc of f_y with an arc of g over "nearby" rational points (where "nearby" means closer than any other event of $\{f_y, g\}$), and this then is the g-arc intersecting f. So our algorithm is this:

Compute $R := \mathrm{res}(f_y, g, y)$. If $R = 0$, signal "coprimality violated" and abort. If $R(x_i) \neq 0$, signal "covertical events" and abort.

Refine $]r_i, r_{i+1}[$ to $]r_-, r_+[\ni x_i$ such that $[r_-, r_+]$ contains no zero of R except x_i and no zero at all of $\mathrm{res}(f_y, f_{yy}, y)$.

Compare the arcs of f_y and g over $x = r_-$ and $x = r_+$:
If there is a transposition of the relevant arc of f_y with some arc of g,
then that arc of g contains (x_i, y_i);
else signal "no intersection in acnode visible" and abort.

4.2.7 Intersection singular-singular

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

If we have a rational representation for x_i, factoring $f(x_i, y), g(x_i, y) \in \mathbb{Q}[y]$ by multiplicities yields rational representations for all points of $\{f, g\}$ over x_i, making the analysis trivial.

From $h := \gcd(R_{fgm_i^{(fg)}}, R_{fm_i^{(f)}}, R_{gm_i^{(g)}})$, we can always obtain a rational representation, because necessarily $\deg(h) = 1$ if the conditions on $\{f, g\}$ are met. This is seen as follows: If $\deg(h) \geq 2$, 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 noncoverticality condition requires $s_j = s'_j = s''_j$ for $j \in \{1, 2\}$. The line connecting these two singular intersection points is a component of both f and g, contradicting their coprimality.

Coprimality has already been checked at this point, so we may conclude from $\deg(h) > 1$ that the noncoverticality condition has been violated.

4.3 Curves, curve pairs, and slices

We have implemented the geometric analyses as member functions of objects that represent curves and curve pairs. A call to such a member function performs the analysis only as far as necessary. All information is cached, so that a costly computation is never done twice. A global table avoids multiple construction of a curve pair object for $\{f, g\}$ even if the pair $\{f, g\}$ is considered at several unrelated occasions.

A curve pair object presents the information on the relative position of arcs over each cell in the decomposition (16) of the x-axis in the following unified form: Given the index i of an event x-coordinate x_i or of an interval $]x_i, x_{i+1}[$ between events, return a *slice*, that is a pair of tables such that the n-th entry in the table for f gives the arc number of the n-th f-arc among all arcs of f and g, and similarly for g (see Figure 8). Comparing arc numbers is equivalent to comparing y-coordinates. In particular, intersections of f and g are reflected by equal arc numbers of an f-arc and a g-arc. Instead of an index i, one can also specify an algebraic x-coordinate ξ identifying one cell of (16), or the special x-coordinate values $\pm\infty$, which we take to refer to the first and last interval, respectively.

4.4 Choosing Coordinates

We imposed certain conditions on curves and curve pairs regarding the choice of a coordinate system; or more precisely, on the choice of the vertical axis. In this section, we will first derive a rough constant upper bound on the number

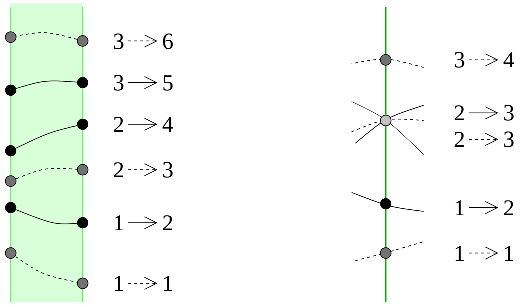

Fig. 8. Slices of a curve pair: over an interval between events (left); over an intersection point which is a crunode of one of the curves (right).

of forbidden directions of the y-axis per curve or curve pair. To do so, we rephrase these conditions on the coordinate system as conditions that certain lines (which are well-defined without reference to a specific coordinate system) shall be non-vertical. Afterwards, we discuss how to change coordinates if a curve or curve pair analysis aborts because of a condition violation.

4.4.1 Forbidden vertical directions

Let us first consider the conditions for a single curve f. The items below match the items in the list of conditions given in §4.1, except for the squarefreeness condition.

- A curve f is y-regular iff its highest-order terms are not divisible by x. The highest-order terms form a homogeneous polynomial of degree $\deg(f)$ which decomposes into linear factors over $\mathbb{C}$. These are the *complex asymptotes* of f. We obtain: f is y-regular iff none of its complex asymptotes is vertical. A cubic curve has at most 3 distinct complex asymptotes.
- No two points of $\mathrm{V}_{\mathbb{C}}(f) \cap \mathrm{V}_{\mathbb{C}}(f_y)$ are covertical by Lemma 20.
- A curve f has no vertical flexes iff the unique tangents in all flexes are non-vertical. A cubic curve has at most 9 flexes [40].
- The non-verticality of tangents in the singularities of a y-regular curve f excludes at most 2 further directions: For an irreducible cubic we demand the non-verticality of the two tangents in the singularity. For a cubic consisting of a conic and a line intersecting in two distinct singularities we demand non-verticality of the respective tangents to the conic component. A tangent to a line component is the line itself which is non-vertical by y-regularity.

This yields an upper bound of **14** on the number of forbidden directions for a single curve.

Now consider a curve pair $\{f, g\}$ and the conditions imposed in §4.2 on relations between the curves, except coprimality.

- The noncoverticality of the ≤ 9 distinct complex intersection points is equivalent to the nonverticality of the $\leq \binom{9}{2} = 36$ lines joining any two of them.

- The noncoverticality of intersection points to one-curve events of f is equivalent to the nonverticality of any line h that both contains an intersection point $p \in f \cap g$ and has a multiple intersection point $q \neq p$ with f. This is because a multiple intersection of h and f at q, that is $\operatorname{mult}(q; h, f) \geq 2$, means that h is a tangent to f at q or that f is singular at q (Proposition 17).

  The noncoverticality of one-curve events of f and one-curve events of g is equivalent to the nonverticality of any line h that has points of multiple intersection with both f and g, by the same argument.

  For both parts, we will bound the respective number of lines below, after introducing the necessary tools.
- There are ≤ 9 non-singular intersection points, each with a unique tangent on f and on g, respectively. These points are not extreme iff their $\leq 2 \cdot 9 = 18$ tangents are non-vertical.
- The Jacobi curve $J = f_x g_y - f_y g_x$ is well-defined independent of a specific choice of coordinates (Proposition 22). It has at most $\deg(J) \leq 4$ distinct complex asymptotes whose nonverticality is equivalent to y-regularity.

  There are $k \in \{0, \ldots, 4\}$ non-singular intersections v of f and g with $\operatorname{mult}(v; f, g) = 2$. For the noncoverticality of Jacobi intersections, no line through any of them that also contains another intersection of J with either f or g is allowed to be vertical. There are at most $\deg(f)\deg(J) - k \leq 12 - k$ of those other intersections with f; same for g. This yields $\leq 2k(12-k) \leq 64$ forbidden lines.

  For the absence of Jacobi one-curve events, no line through any of the ≤ 4 points v that also has a multiple intersection with J is allowed to be vertical (as before because of Proposition 17). We will bound the number of such lines below.
- Consider the acnode v of a complex triangle f. The discriminant $D(x, y)$ of the quadratic part of $f(x+r, y+s)$ is $D = f_{xy}^2 - f_{xx} f_{yy}$ by Taylor's formula. Since v is an acnode, $D(v) \neq 0$. Notice that D is equal to the Jacobi curve $\det(\nabla f_y, \nabla f_x)$. Hence $\nabla f_x(v)$ and $\nabla f_y(v)$ are linearly independent.

  Partial differentiation after a change of coordinates induced by some invertible matrix $A = (a_{ij})_{i,j}$ yields the curve $(((f \circ A)_y) \circ A^{-1})(v) = a_{12} f_x(v) + a_{22} f_y(v)$ with gradient $a_{12} \nabla f_x(v) + a_{22} \nabla f_y(v)$. Hence just one choice of a new vertical direction $(a_{12} : a_{22})$ makes $\nabla f_y(v)$ parallel to $\nabla g(v)$; for all other choices they are non-parallel, so that coprimality of g and the irreducible conic f_y as well as $\operatorname{mult}(v; f_y, g) = 1$ are certain.

To obtain estimates on the number of lines h fulfilling a condition of the form "h has a multiple intersection point with f (and we don't care where)", we lift the scene to the complex projective plane by homogenizing $f(x, y)$ to $F(x, y, z) = z^{\deg(f)} f(x/z,\, y/z)$, and we consider the set of all complex-projective lines H that have a multiple intersection point with F. Their duals H^* form an algebraic curve $\tilde{F}$ in dual space (unless F has a line component, but we leave out this special case for brevity), which has degree $\deg(F)(\deg(F)-1)$

[14, pp. 252+]. For our setting, this means $\deg(\tilde{F}) \leq 6$. (Removing the line components from $\tilde{F}$, which reflect the intersections in singularities, gives the well-known *dual curve* F^* of F [14, ibid.] [76, V.8] [40, 16.6].)

With these notions, we can now dualize "a line H through p having a multiple intersection point with curve F" to "an intersection point H^* of line p^* and curve $\tilde{F}$". This yields an upper bound $\deg(\tilde{F})$ on the number of such lines H.

Let us proceed to bound the number of lines H having a multiple intersection with F outside some fixed non-singular point p of F. Let T be the tangent to F at p. Recall that, for degree reasons, a line cannot have more than one multiple intersection with a cubic. Hence "a line H through p having a multiple intersection point with curve F outside p" is equivalent to "a line $H \neq T$ through p having a multiple intersection point with curve F". We can dualize the latter to "an intersection point H^* of line p^* and curve $\tilde{F}$ distinct from T^*". It is known [14, p. 255] that the intersection of $\tilde{F}$ and p^* at T^* has multiplicity 2. Hence we have an upper bound of $\deg(\tilde{F}) - 2$ on the number of lines H.

Finally we can dualize "a line H having points of multiple intersection with both F and G" to "an intersection point of $\tilde{F}$ and $\tilde{G}$". Bézout's Theorem (p. 56) implies that no more than $\deg(\tilde{F})\deg(\tilde{G})$ such lines H exist.

Returning to the original question, we obtain the following bounds:

- Through each of the ≤ 9 intersection points p of f and g, there are $\leq 3 \cdot 2 - 2 = 4$ lines that have a multiple intersection point with f outside p; same for g. This forbids $\leq 2 \cdot 4 \cdot 9 = 72$ directions of lines.
- There are $\leq (3 \cdot 2)^2 = 36$ lines that have points of multiple intersection with both f and g.
- Through each of the ≤ 4 non-singular intersection points of f and g with multiplicity 2, there are $\leq 4 \cdot 3 = 12$ lines that have a multiple intersection with the Jacobi curve J. This forbids $\leq 4 \cdot 12 = 48$ directions of lines.

These three items forbid $\leq 72 + 36 + 48 = 156$ directions. Together with the $\leq 36 + 18 + 68 + 1 = 123$ forbidden directions from the preceding list, we obtain that no more than $156 + 123 = \mathbf{279}$ directions of a y-axis are forbidden per analyzed curve pair.

4.4.2 Random Shearing

Since there are infinitely many possible y-axes, a random choice will pick one that is permissible for all curves and curve pairs analyzed during the algorithm with probability 1. Hence our strategy for finding a permissible coordinate system (one that is *generic* w. r. t. our conditions) is this: Shear the input scene with a random shearing parameter $r \in \mathbb{Q}$ and run the algorithm. Check the conditions along the way. If a violation is detected, pick a new r and restart. A *shear* is an invertible linear map $S_r\colon (x, y) \mapsto (x + ry, y)$ for a fixed $r \in \mathbb{Q}$. It leaves the x-axis fixed and tilts the y-axis.

From which range should the algorithm choose r? We do not recommend to compute an a priori bound on the number of forbidden directions and define a range larger than that, because most forbidden directions will lie outside that range, being irrational or rational with large denominator, and choosing from a large range comes at a price: A value r of binary encoding length s will increase the coefficient size of a curve f by a factor of $s \deg(f)$. Instead, start with a small range and increase its size with the number of past failures.

4.4.3 Shearing back

We will apply curve and curve pair analyses to implement predicates for arrangement computation in Section 5. We propose to choose one global generic coordinate system. Transforming input curves from the original into the generic (sheared) system is not a problem. Analyzing curve and curve pair geometry in the sheared system has been solved above. Topological data like the graph representing the arrangement is valid in any coordinate system. However, transforming event point coordinates back from the sheared into the original coordinate system creates some complications.

Our algorithms for curve and curve pair analysis express y-coordinates in an implicit fashion as arc numbers. This entails a representation of event points that is not amenable to applying the inverse shear, because there are no explicit y-coordinates.

Instead, we propose to obtain approximations of coordinates: Use numerical methods (with or without guarantees) for solving univariate polynomials to approximate event point coordinates in the sheared coordinates, then transform back. Numerical approximation is made easier by the fact that the algebraic computations in the analysis of curves help to avoid solving for multiple zeroes. (The defining polynomials for x-coordinates ξ are explicitly made square free. The y-coordinate of a one-curve event, if not known rationally, is definable with $f_y(\xi, y)$ instead of $f(\xi, y)$.)

Generating correct topology and arbitrarily accurate numerical point coordinates is consistent with the idea of Exact Geometric Computation (EGC) [81]. Using a generic coordinate system considerably reduces the number of cases distinguished in the geometric analyses. However, there are two disadvantages: First, the sheared back result itself is not represented exactly, complicating further operations on it. Second, introducing a new curve may put the formerly generic coordinate system in violation of a position condition, requiring the whole computation to restart with a new random choice of coordinates;

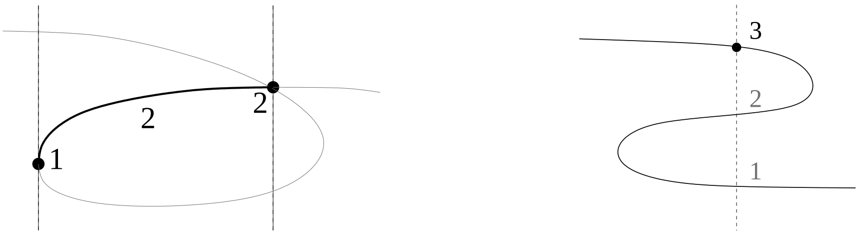

Fig. 9. Representations for a sweepable segment (left) and a point (right).

hence an incremental algorithm must retain history data to handle such an incident.

5 Arrangement Computation

We show how to perform a Bentley-Ottmann-like sweep [9] of curves, in the complete formulation from LEDA [62, ch. 10.7], based on the analysis of curves and curve pairs. We emphasize that the following presentation is not restricted to cubic curves, but works for all curves for which we can provide analyses similar to §4.1 and §4.2 (essentially, a cylindrical algebraic decomposition with information on adjacencies and intersection multiplicities).

As input, we accept a set of curves. As output, our method computes a planar map labelled with points (including auxiliary points like extreme points) and input curves, representing the arrangement.

In a preprocessing step, every input curve f is broken into sweepable *segments* such that each segment s has no one-curve event in its interior and such that all points in the interior of s have the same arc number i, meaning that β is the i-th real root of $f(\alpha, y)$ for every $(\alpha, \beta) \in \text{int}(s)$.

A segment is represented by its endpoints, its supporting curve, and its respective arc numbers in the interior and at the endpoints (Fig. 9 (left)). (This representation also allows user-defined segments that are subsets of the curve's sweepable segments.) A point is represented by an x-coordinate, a supporting curve and an arc number (Fig. 9 (right)). To represent arcs extending to infinity, we allow the special values $-\infty$ and $+\infty$ for the x-coordinate.

We outline the generalized Bentley-Ottmann algorithm for completeness (cf. Figure 10). 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

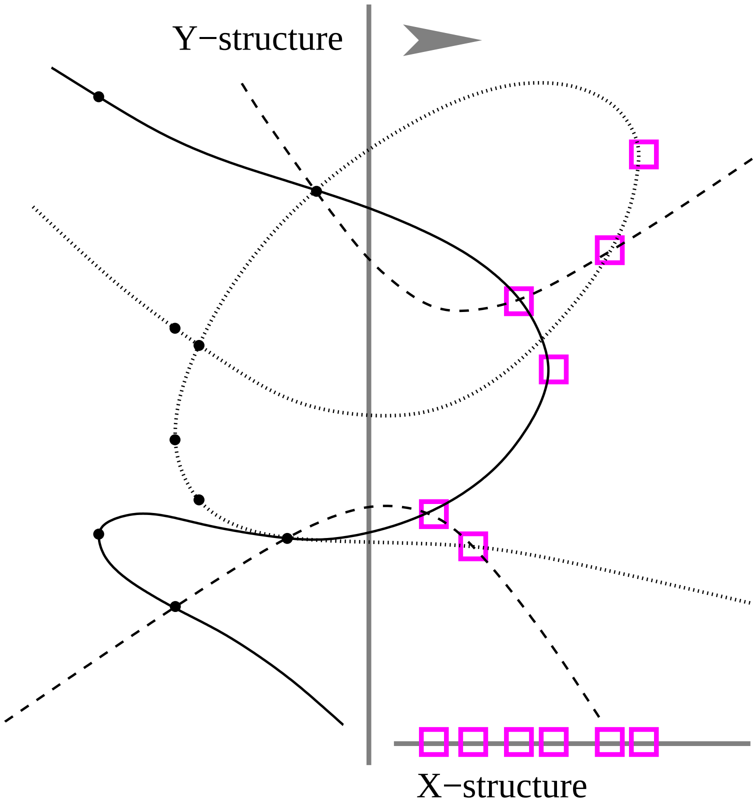

Fig. 10. Illustration of the Bentley-Ottmann sweep line algorithm.

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. Covertical event points are handled 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 the next event point from the X-structure. Find the segments involved in the event by locating the event point in the Y-structure, exploiting its order.
(Requires: Comparison of y-coordinates of point and 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 in-

tersection (explained below in §5.1) such that the new order reflects the situation right of the event.
(Requires: Segment overlap test. — 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 common point.)

(5) Add intersections of newly adjacent segments to the X-structure, obeying its ordering.
(Requires: Computation of segment intersection points. — Lexicographic comparison of event points.)

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.[7]

In summary, we need the following predicates and constructions:

Lexicographic comparison of event points. If comparing x-coordinates does not break the tie, slice the two supporting curves at their common x-coordinate to compare the supporting arcs.

Comparison of y-coordinates of point and segment. Decide this using the slice of the two supporting curves at the point's x-coordinate.

Comparison of segment y-order right of common point. Inspect the slice of the two supporting curves over the interval right of the intersection.

Segment overlap test. Since different curves are required to be coprime, which is checked when the resultant of the corresponding curve pair is computed, segments overlap if and only if they are non-trivial, their supporting curves are the same, their interior arc numbers are the same, and their x-ranges overlap.

Computation of segment intersection points. Analyze the two supporting curves over the intersection of the segments' x-ranges. For each two-curve event in that range, determine from the slice whether the supporting arcs of the segments coincide.

Intersection multiplicity of non-overlapping segments. This is the multiplicity of the corresponding root of $\mathrm{res}(f, g, y)$ by noncoverticality of intersections.

[7] Our representation of a point involves only one curve but an explicit x-coordinate, so that this view is justified, even though, for example, the comparison of two intersection points can be seen as involving four curves, i. e., the two pairs of curves creating the intersections.

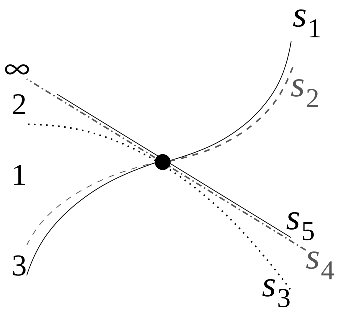

Fig. 11. Segments changing their y-order in an intersection point. The numbers on the left are the intersection multiplicities of adjacent segments.

5.1 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. We present the approach from [11] formalized with the notions of Section 3.

In the reordering step, depicted in Figure 11, only segments containing the event point in their interior are involved. Let us call them $s_1, \ldots, s_k$, 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

$$y = \varphi_i(x) = \sum_{d=1}^{\infty} a_d^{(i)} x^d \qquad (20)$$

after translating the event point to $(0, 0)$. The coefficients of the implicit functions determine the y-order of segments just left and just right of the intersection.

Proposition 24 *With notation as above:*
Segment s_i lies below segment s_j right of the intersection iff

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

Segment s_i lies below segment s_j left of the intersection iff

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

(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, a finite $m = \min\{d \mid a_d^{(i)} \neq a_d^{(j)}\}$ exists, and $\varphi_i(x) - \varphi_j(x) = (a_m^{(i)} - a_m^{(j)})x^m + \ldots$ is negative for small positive x iff $a_m^{(i)} < a_m^{(j)}$. □

By Proposition 18, the quantity m considered in the proof for non-overlapping segments is precisely the intersection multiplicity of the segments' supporting curves in the sense of §3.1. Incorporating the case of overlap, we define the intersection multiplicity of segments s_i and s_j as $\min(\{d \mid a_d^{(i)} \neq a_d^{(j)}\} \cup \{\infty\})$.

When we come to process the intersection of $s_1, \ldots, s_k$, intersection multiplicities will typically not have been computed for all $\frac{1}{2}k(k-1)$ pairs of segments. For $1 \leq i < k$ let $m_i \in \{1, 2, 3, \ldots, \infty\}$ denote 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. The following result allows us to infer all other intersection multiplicities from them.

Proposition 25 *With notation as above, the intersection multiplicity of s_i and s_j, $1 \leq i < j \leq k$, is $\min\{m_i, \ldots, m_{j-1}\}$.*

Proof. By induction on j. The base case $j = i + 1$ is clear. For the inductive step from j to $j+1$, let $m = \min\{m_i, \ldots, m_{j-1}\}$ be the intersection multiplicity of s_i and s_j. For $m_j = \infty$, the claim is clear. Otherwise, distinguish three cases:

The first case is $m > m_j$. It holds that $a_d^{(j+1)} = a_d^{(j)} = a_d^{(i)}$ for $d < m_j$ and $a_d^{(j+1)} \neq a_d^{(j)} = a_d^{(i)}$ for $d = m_j$, so that the intersection multiplicity of s_i and s_{j+1} is $m_j = \min\{m, m_j\}$.

For $m < m_j$, we have equality for $d < m$ 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 24 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 way of performing step (3) (cf. [11]):

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 $m = M, M-1, \dots, 1$, take all maximal subsequences $s_i, s_{i+1}, \dots, s_j$ with the property $m_i, m_{i+i}, \dots, m_{j-1} \geq m$ 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 $n = \min\{m_i, \dots, m_{j-1}\}$, 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 25, n is their intersection multiplicity. By Corollary 19, 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 $s_i, s_{i+1}, \dots, s_j$, 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 segment reordering is $O(M \cdot k)$. In our context M is at most 10, according to Bézout's Theorem (p. 56) and M being even.

6 Runtime Analysis

The runtime of the sweep in the Real RAM model remains the known $O((n+s)\log(n+m)+m)$ [62, ch. 10.7], where n is the number of curves, s the number of nodes, and m the number of edges in the resulting planar map. Here it is essential that we reorder all segments passing through an intersection point in linear time.

We consider the effect of shearing on the runtime. We show in §4.4.1 that we have only a constant number of forbidden directions for each analyzed pair of curves. We conclude that a random choice among quadratically many directions will lead to an expected constant number of shears. Since bit-complexity is of interest, we actually bias the choice towards directions of small bit size representations.

Besides the Real RAM model, the bit-complexity is of obvious importance here. However, a complete worst-case analysis is impractical (see the number of case distinctions), and furthermore, we expect no promising result from the known separation bounds that we would need to apply for the (cascaded) root isolations [19].

Instead, we emphasize that our approach is not tied to the worst case. Our methods benefit whenever a particular instance does not require the isolating intervals to approach the separation bounds limit but can stop earlier. Not only does the iteration stop earlier, e. g., in the Descartes Method, but also the bit complexity of the interval boundaries becomes smaller and subsequent steps are faster. Even more important, we do not use the separation bound approach to detect equality between our algebraic numbers. This is in contrast, e. g., to the `leda::real` and `CORE::Expr` implementations, where the equality test is the most costly decision. Instead we detect equality of two algebraic numbers by finding a common factor of their defining polynomials with a root in the appropriate interval. This is much faster than refining the intervals to their separation bounds.

On top of this, we use modular arithmetic to quickly filter out gcd computations in cases of coprime defining polynomials (§2.4.2). The same idea helps to speed up factorization by multiplicities (§2.4.1).

In summary, we argue that the worst-case analysis of the bit complexity would be non-representative for our approach. Instead, we illustrate its efficiency with the experiments in the next section.

An alternative to the sweep-line algorithm is the randomized incremental construction because of its better asymptotic runtime. We can realize the necessary predicates with our approach. However, since we do not simply determine signs of polynomial expressions, it is not clear whether the lower degree of predicates[8] for the linear case carries over to our setting.

7 Implementation and Experiments

We have implemented our method in the project Exacus, *Efficient and Exact Algorithms for Curves and Surfaces.*[9] Exacus is a collection of C++ libraries; NumeriX provides the algebraic and numerical foundations, SweepX provides the generic sweep-line algorithm independent of the curve type, and CubiX provides a full implementation of the curve and curve pair analyses

[8] The degree of a predicate is the degree of the polynomial expression in the input data whose sign yields the result of the predicate.

[9] `http://www.mpi-inf.mpg.de/projects/EXACUS/`

of cubic curves and combines them with the sweep-line algorithm in a demo program. The demo visualizes curves independent of arrangement computation with a subdivision method by Taubin [75] (see [61] for a survey of other options). This is helpful for demonstration and debugging purposes and also for interactive creation of test instances.

Exacus follows the *generic programming paradigm* [64]. Our C++ implementation is based upon design experience gained with C++ templates in this paradigm with the *Standard Template Library*, Stl [7], and the *Computational Geometry Algorithms Library*, Cgal [33,49]. One example for a generic reusable component in Exacus is the polynomial class template with the flexibility of different coefficient types and efficiency at the same time. More on the software structure and design can be found in [10].

We offer three series of benchmarks. Firstly, there is a series of *random* sets of n cubic curves. Each curve f is defined by interpolation through 9 points chosen uniformly at random from a set of $9n$ random points on the $\{-128, \ldots, 127\}^2$ integer grid. Every interpolation point results in a homogeneous linear condition on the 10 unknown coefficients of f, so that generically 9 conditions determine the equation of a curve uniquely, up to a constant factor. For each input size n, we have generated an odd number of candidate input data sets and picked the one with median average running time for inclusion in our benchmark.

Secondly, there is a series of *degenerate* instances. It is obtained in a similar fashion, except that

(1) There are only 54 interpolation points.
(2) At the first interpolation point of each curve, we demand with probability 0.2 that not only f but also f_x and f_y vanish, making this point a rational singularity (yielding 2 additional linear conditions).
(3) For each interpolation point p, we pick random values for slope m_p and curvature κ_p. Whenever a curve f is interpolated through p, we make its slope equal to m_p (yielding one additional linear condition) and, with probability $\sqrt{0.5}$, we also make its curvature equal to κ_p (yielding a further linear condition).

The result is an arrangement of curves with (1) vertices of high degree, (2) curves with singularities, (3) two- and threefold intersections.

Thirdly, there is a series with *coefficient growth*: We take the $n = 60$ instances from the preceding *random* and *degenerate* series, scale each interpolation point p by a factor $s = 100$, $10\,000$, or $1\,000\,000$, and then perturb it to $sp + \varepsilon$ with offset vector ε chosen randomly from $\{-10, \ldots, 10\}^2$. (All occurrences of an interpolation point are mapped in the same way such as to preserve degeneracies.) This increases the bit size of the curves' coefficients but preserves the

Series	n	segments	nodes	half-edges	bits	time
random	30	226	2933	11038	99	6.1 s
	60	454	11417	44440	99	25.1 s
	90	680	26579	104474	100	62.2 s
	120	940	46117	181978	100	114.8 s
	150	1226	71594	283114	99	180.7 s
	180	1460	102298	405312	101	260.2 s
	200	1554	126278	500888	101	322.5 s
degenerate	30	243	2313	8604	116	11.1 s
	60	534	7627	29284	116	40.8 s
	90	722	17983	70378	120	95.6 s
	120	1027	31504	123814	114	168.4 s
	150	1292	48362	190698	116	258.9 s

Table 1
Running times in seconds for the *random* and *degenerate* benchmark instances.

combinatorial structure of the arrangement (if s is sufficiently large), allowing us to measure the increased cost of arithmetic as a function of bit sizes.

We report average running times for arrangement computations measured on a 1.2 GHz Pentium IIIM system with 512 KB of cache running Linux. The executable was compiled with g++ 3.1. LEDA 4.4 was used for the exact number types and the internal data structures of our sweep code. All benchmark instances are computed in the original coordinate system; that is, they have not been transformed with a random shear.

Table 1 reports the results for the *random* and *degenerate* instances. Each row states number of input curves, total number of segments after splitting of input curves, number of nodes and half-edges in the resulting graph, the average bit length of a curve's longest coefficient, and the average running time in seconds. The running time is dominated by curve pair analyses (§4.2). The fraction of time spent on curve analyses (§4.1) is well below one second even for the largest instances.

The plot in Figure 12 shows the running times as a function of the number of computed half-edges (output complexity). In accordance with the theoretical analysis, the output complexity looks almost linear. However, the output size is quadratic in the number of curves, as for the straight-line case.

The modular filter for gcd computations has been found to accelerate the "random 30" instance by a factor of 9. This filter and the caching of curve

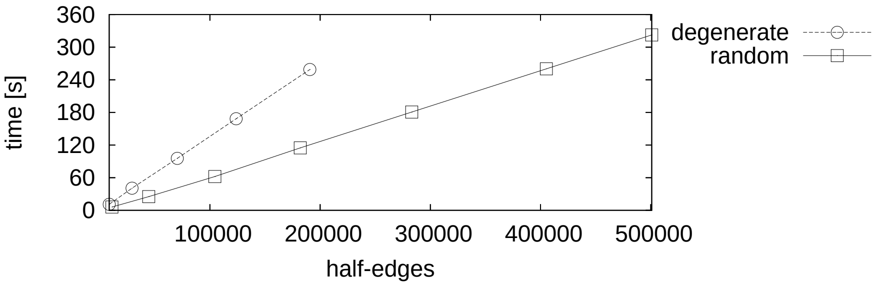

Fig. 12. The running times from Table 1 as a function of the computed half-edges.

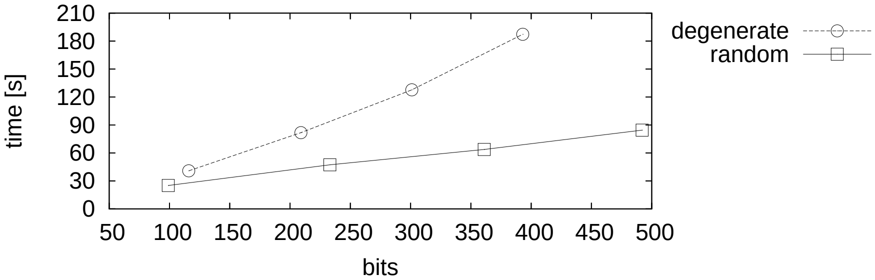

Fig. 13. The running times from Table 2 as a function of average maximal coefficient length.

and curve pair analyses for repeated use in predicate evaluations are important sources of efficiency in our implementation.

As dominant reasons for slowdown in the degenerate instances we see: Multiple intersections of f, g cause multiple zeroes in $\mathrm{res}(f, g, y)$. High-degree vertices correspond to equality of event point x-coordinates. Both phenomena entail the computation of gcds that are avoided by the modular filter in the generic case. The analysis of a tangential intersection requires the consideration of a Jacobi curve, involving the refinement of an isolating interval against three additional resultants of degree 12.

Table 2 and Figure 13 show the results of the *coefficient growth* benchmark. In this setup, the increase in running time is bounded by the growth of running time for the exact arithmetic, especially multiplication. Our experiments used LEDA with the $O(N^{\log_2 3})$ Karatsuba multiplication. This superlinear growth is well-visible for the degenerate instances, and indeed they invoke more symbolic computations (such as gcds that are otherwise avoided and additional resultants for Jacobi curves). For the random instances, the superlinear term is less pronounced, reflecting the fact that the coefficients of curves and resultants grow, whereas the interval boundaries in root isolation and comparison do not, so that in these parts of the algorithm only one of the two operands of multiplication grows.

Finally, we can report on a brief comparison with arrangement computation

Instance	segments	nodes	half-edges	bits	time
random 60	454	11417	44440	99	25.1 s
with $s = 10^2$	454	11437	44520	233	47.3 s
with $s = 10^4$	454	11417	44440	361	63.8 s
with $s = 10^6$	454	11417	44440	492	84.5 s
degenerate 60	534	7627	29284	116	40.8 s
with $s = 10^2$	534	7639	29332	209	81.8 s
with $s = 10^4$	534	7627	29284	301	127.7 s
with $s = 10^6$	534	7627	29284	393	187.3 s

Table 2
Running times in seconds resulting from slight perturbations that increase coefficient sizes but keep the geometry (almost) fixed.

for straight-line segments in LEDA 4.4.1. We have created sets of line segments defined by pairs of points chosen randomly on the $\{-128, \ldots, 127\}^2$ integer grid such that the resulting arrangement has around 105 000 half-edges, similar to the "random 90" instance above. Computing such a straight-line arrangement using unfiltered rational arithmetic is on average about 10 times faster than our results for cubics. Switching on floating-point filtering accelerates the straight-line arrangements by a further factor of 6 to 7. Recall that our code does not contain floating-point filtering at this time. Considering the elaborate algebraic methods deployed for cubics, one order of magnitude difference between unfiltered implementations looks quite encouraging to us.

8 Conclusion

We have reduced systematically – in theory and in the form of generic C++ library code – all predicates necessary for arrangement computation of algebraic curves to the analysis of curves and curve pairs, and to the handling of x-coordinates. For the specific case of degree ≤ 3, we have combined techniques from symbolic computation with geometric observations to design and implement these basic operations efficiently. Important parts of this design are directly applicable to curves of higher degree, but others – in particular in the treatment of singularities – are not.

To achieve completeness, we choose a generic coordinate system by shearing the input data. Although clearly useful to curb case distinctions, changing coordinates also has its disadvantages as discussed in §4.4.3. Most notably, we cannot apply the inverse transformation on our exact implicit representation.

We have successfully deployed the modular filter (§2.4.1, §2.4.2) as a fast probabilistic non-zero test. The impact of guaranteed floating-point methods for filtering, in particular methods for finding isolating intervals for roots, remains to be investigated.

As detailed in the introduction, the techniques used here are different from those used previously for conic arcs. When we compared the CubiX implementation described here with the original ConiX implementation from [11], CubiX proved to be several times faster for arrangements of conics. We regard this as encouraging evidence for the applicability of the combination of techniques used here. Furthermore, our understanding of how to implement the geometric predicates, which information to cache, etc., has improved considerably. The lessons learned here have been back-ported by Eric Berberich to ConiX, which delivers the same performance now.

Our method for arrangement computation extends directly to arrangements of user-defined segments of cubic curves and thus to boolean operations on curvilinear polygons bounded by such segments.

Acknowledgements

The authors acknowledge the support by the following other members of the Exacus team (past and present): Eric Berberich, Michael Hemmer, Susan Hert, Kurt Mehlhorn, Joachim Reichel, Susanne Schmitt, and Evghenia Stegantova. The authors acknowledge helpful conversations with M'hammed El Kahoui.

References

[1] J. Abdeljaoued, G. M. Diaz-Toca, L. Gonzalez-Vega: "Minors of Bezout Matrices, Subresultants and the Parameterization of the Degree of the Polynomial Greatest Common Divisor". *Internat. J. of Computer Math.* **81** (2004), pp. 1223–1238.

[2] P. K. Agarwal, M. Sharir: "Arrangements and their applications". In J.-R. Sack, J. Urrutia (eds.): *Handbook of Computational Geometry*, pp. 49–119. Elsevier, 2000.

[3] A. G. Akritas: "There is no 'Uspensky's Method' ". *Proc. 5th ACM Symp. on Symbolic and Algebraic Computation* (SYMSAC 1986), pp. 88–90, ACM, New York, 1986.

[4] A. G. Akritas: *Elements of Computer Algebra.* Wiley, New York, 1989.

[5] D. S. Arnon, G. E. Collins, S. McCallum: "Cylindrical Algebraic Decomposition I: The Basic Algorithm". *SIAM Journal on Computing* **13:4** (1984) pp. 865–877.
Reprinted in: B. F. Caviness, J. R. Johnson (eds.), *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pp. 136–151, Springer, 1998.

[6] D. S. Arnon, G. E. Collins, S. McCallum: "Cylindrical Algebraic Decomposition II: An Adjacency Algorithm for the Plane". *SIAM Journal on Computing* **13:4** (1984) pp. 878–889.
Reprinted in: B. F. Caviness, J. R. Johnson (eds.), *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pp. 152–165, Springer, 1998.

[7] M. H. Austern: *Generic Programming and the STL*. Addison-Wesley, 1998.

[8] S. Basu, R. Pollack, M.-F. Roy: *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics, vol. 10. Springer, Berlin, 2003. See also `http://www.math.gatech.edu/~saugata/bpr-posted1.html`

[9] J. L. Bentley, T. A. Ottmann: "Algorithms for reporting and counting geometric intersections". *IEEE Trans. Comput.*, **C-28:9** (1979) pp. 643–647.

[10] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, L. Kettner, K. Mehlhorn, J. Reichel, S. Schmitt, E. Schömer, N. Wolpert: "EXACUS: Efficient and Exact Algorithms for Curves and Surfaces". *Proc. 13th European Symp. on Algorithms* (ESA 2005), pp. 155–166, Springer LNCS 3669, Berlin, 2005.

[11] 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. 10th European Symp. on Algorithms* (ESA 2002), pp. 174–186, Springer LNCS 2461, Berlin, 2002.

[12] P. Bikker, A. Yu. Uteshev: "On the Bézout Construction of the Resultant". *J. Symbolic Computation* **28** (1999), pp. 45–88.

[13] R. Bix: *Conics and Cubics: A Concrete Introduction to Algebraic Curves*, Springer UTM, New York, 1998.

[14] E. Brieskorn, H. Knörrer: *Plane Algebraic Curves*. Birkhäuser, Basel, 1986. German original: *Ebene algebraische Kurven*, Birkhäuser, Basel, 1981.

[15] 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.

[16] W. S. Brown: "The Subresultant PRS Algorithm". *ACM Trans. Math. Software* **4** (1978), pp. 237–249.

[17] W. S. Brown, J. F. Traub: "On Euclid's Algorithm and the Theory of Subresultants". *J. ACM* **18** (1971), pp. 505–514.

[18] 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.

[19] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, S. Schmitt: "A separation bound for real algebraic expressions". *Proc. 10th European Symp. on Algorithms* (ESA 2001), pp. 254–265, Springer LNCS 2161, Berlin, 2001.

[20] J. Canny: *The Complexity of Robot Motion Planning.* ACM–MIT Press Doctoral Dissertation Award Series. MIT Press, Cambridge, MA, 1987.

[21] H. Cohen: *A Course in Computational Algebraic Number Theory.* Springer GTM 138, Berlin, 1993.

[22] G. E. Collins: "Subresultants and Reduced Polynomial Remainder Sequences". *J. ACM* **14** (1967), pp. 128–142.

[23] G. E. Collins: "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition". *Proc. 2nd GI Conf. on Automata Theory and Formal Languages*, pp. 134–183, Springer LNCS 33, Berlin, 1975.
Reprinted with corrections in: B. F. Caviness, J. R. Johnson (eds.), *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pp. 85–121, Springer, 1998.

[24] G. E. Collins, A. G. Akritas: "Polynomial Real Root Isolation Using Descartes' Rule of Signs". *Proc. 3rd ACM Symp. on Symbolic and Algebraic Computation* (SYMSAC 1976), pp. 272–275, ACM, New York, 1976.

[25] D. Cox, J. Little, D. O'Shea: *Ideals, Varieties, and Algorithms*, 2nd ed. Springer UTM, New York, 1997.

[26] O. Devillers, A. Fronville, B. Mourrain, M. 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.

[27] G. M. Diaz-Toca, L. Gonzalez-Vega: "Various New Expressions for Subresultants and Their Applications". *Applic. Algebra in Engin., Commun. and Computing* **15** (2004), pp. 233–266.

[28] A. Eigenwillig: *Exact Arrangement Computation for Cubic Curves.* Master's thesis, Saarland University, Saarbrücken, Germany, 2003.

[29] A. Eigenwillig, L. Kettner, E. Schömer, N. Wolpert: "Complete, Exact, and Efficient Computations with Cubic Curves". *Proc. 20th Ann. Symp. on Comp. Geom.* (SCG 2004), pp. 409–418, ACM, New York, 2004.

[30] M. El Kahoui: "An elementary approach to subresultants theory". *J. Symbolic Computation* **35** (2003), pp. 281–292.

[31] I. Emiris, A. Kakargias, S. Pion, M. Teillaud, E. Tsigaridas: "Towards an Open Curved Kernel". *Proc. 20th Ann. Symp. on Comp. Geom.* (SCG 2004), pp. 438–446, ACM, New York, 2004.

[32] I. Emiris, E. Tsigaridas: "Comparing Real Algebraic Numbers of Small Degree". *Proc. 12th European Symp. on Algorithms* (ESA 2004), pp. 652–663, Springer LNCS 3221, Berlin, 2004.

[33] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, S. Schönherr: "On the Design of CGAL, a Computational Geometry Algorithms Library". *Software – Practice and Experience* **30** (2000) pp. 1167–1202.

[34] E. Flato, D. Halperin, I. Hanniel, O. Nechushtan, E. Ezra: "The design and implementation of planar maps in CGAL". *ACM J. of Exper. Algor.* **5** (2000).

[35] W. Fulton: *Algebraic Curves*, Benjamin/Cummings, 1969. Reprint by Addison-Wesley, 1989.

[36] J. von zur Gathen, J. Gerhard: *Modern Computer Algebra.* Cambridge Univ. Press, Cambridge, 1999.

[37] J. von zur Gathen, T. Lücking: "Subresultants revisited". *Theoretical Computer Science* **297** (2003), pp. 199–239.

[38] K. O. Geddes, S. R. Czapor, G. Labahn: *Algorithms for Computer Algebra.* Kluwer, Norwell/Mass., 1992.

[39] I. M. Gelfand, M. M. Kapranov, A. V. Zelevinsky: *Discriminants, Resultants, and Multidimensional Determinants.* Birkhäuser, Boston, 1994.

[40] C. G. Gibson: *Elementary Geometry of Algebraic Curves: an Undergraduate Introduction*, Cambridge University Press, 1998.

[41] R. N. Goldman, T. W. Sederberg, D. C. Anderson: "Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves". *Computer Aided Geometric Design* **1** (1984), pp. 327–356.

[42] L. Gonzalez-Vega, I. Necula: "Efficient topology determination of implicitly defined algebraic plane curves". *Computer Aided Geometric Design* **19** (2002) pp. 719–743.

[43] D. Halperin: "Arrangements". Chapter 24 in J. E. Goodman, J. O'Rourke (eds.): *Handbook of Discrete and Computational Geometry*, 2nd ed. CRC Press, Boca Raton, 2004.

[44] M. Hemmer, L. Kettner, E. Schömer: *Effects of a Modular Filter on Geometric Applications.* Technical Report ECG-TR-363111-01, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, 2004.

[45] H. Hong: "An efficient method for analyzing the topology of plane real algebraic curves". *Mathematics and Computers in Simulation* **42** (1996) pp. 571–582.

[46] X. Hou, D. 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.

[47] J. R. Johnson: "Algorithms for Polynomial Real Root Isolation". In B. F. Caviness, J. R. Johnson (eds.), *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pp. 269–299, Springer, 1998.

[48] 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 1999), pp. 351–359, ACM, New York, 1999.

[49] L. Kettner, S. Näher: "Two Computational Geometry Libraries: LEDA and CGAL". Chapter 65 in J. E. Goodman, J. O'Rourke (eds.): *Handbook of Discrete and Computational Geometry*, 2nd ed. CRC Press, Boca Raton, 2004.

[50] 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 1999), pp. 360–369, ACM, New York, 1999.

[51] D. E. Knuth: *The Art of Computer Programming, vol. 2: Seminumerical Algorithms*, 3rd ed. Addison-Wesley, 1998.

[52] W. Krandick: "Isolierung reeller Nullstellen von Polynomen". In: J. Herzberger (ed.), *Wissenschaftliches Rechnen*, pp. 105–154, Akademie-Verlag, Berlin, 1995.

[53] W. Krandick, K. Mehlhorn: "New Bounds for the Descartes Method". Tech. Rep. DU-CS-04-04, Drexel University, Dept. of Computer Science, `http://www.cs.drexel.edu/static/reports/DU-CS-04-04.html` To appear in *J. Symbolic Computation.*

[54] S. Lang: *Algebra*, 2nd ed. Addison-Wesley, 1984.

[55] C. Li, C. Yap: "A New Constructive Root Bound for Algebraic Expressions". *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms* (SODA 2001), pp. 496–505, ACM, New York, 2001.

[56] T. Lickteig, M.-F. Roy: "Cauchy Index Computation". *Calcolo* **33** (1996), pp. 337–351.

[57] T. Lickteig, M.-F. Roy: "Sylvester-Habicht Sequences and Fast Cauchy Index Computation". *J. Symbolic Computation* **31** (2001), pp. 315–341.

[58] H. Lombardi, M.-F. Roy, M. Safey El Din: "New Structure Theorem for Subresultants". *J. Symbolic Computation* **29** (2000), pp. 663–689.

[59] R. Loos: "Generalized Polynomial Remainder Sequences". In B. Buchberger et al. (eds.): *Computer Algebra. Symbolic and Algebraic Computation*, 2nd ed., pp. 115–137. Springer, Wien, 1983.

[60] R. Loos: "Computing in Algebraic Extensions". In B. Buchberger et al. (eds.): *Computer Algebra. Symbolic and Algebraic Computation*, 2nd ed., pp. 173–187. Springer, Wien, 1983.

[61] 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.

[62] K. Mehlhorn, S. Näher: *LEDA: A platform for combinatorial and geometric computing.* Cambridge Univ. Press, Cambridge, 1999.

[63] M. Mignotte: *Mathematics for Computer Algebra.* Springer, 1992.

[64] D. R. Musser, A. A. Stepanov: "Algorithm-oriented Generic Libraries". *Software – Practice and Experience* **24** (1994) pp. 623–642.

[65] A. M. Ostrowski: "Note on Vincent's Theorem". *Annals of Mathematics, Second Series*, **52** (1950), pp. 702–707, 1950.
Reprinted in: A. Ostrowski, *Collected Mathematical Papers, vol. 1*, pp. 728–733, Birkhäuser, 1983.

[66] P. Pedersen: "Multivariate Sturm Theory". *Proc. 9th Int. Symp. Appl. Algebra, Algeb. Algor. and Error-Corr. Codes* (AAECC-9), pp. 318–332, Springer LNCS 539, 1991.

[67] F. P. Preparata, M. I. Shamos: *Computational Geometry: An Introduction.* Springer, New York, 1985.

[68] G. 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.

[69] F. Rouillier, P. Zimmermann: "Efficient isolation of polynomial's real roots". *J. Computational and Applied Math.* **162** (2004), pp. 33–50.

[70] T. Sakkalis, R. Farouki: "Singular Points of Algebraic Curves". *J. Symbolic Computation* **9** (1990), pp. 405–421.

[71] T. Sakkalis: "The topological configuration of a real algebraic curve". *Bulletin of the Australian Mathematical Society* **43** (1991) pp. 37–50.

[72] S. Schmitt: "The diamond operator: Implementation of exact real algebraic numbers". *Proc. 8th Internat. Workshop on Computer Algebra in Scient. Comput.* (CASC 2005), pp. 355–366, Springer LNCS 3718, Berlin, 2005.

[73] E. Schömer, N. Wolpert: "An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics". *Computational Geometry* **33** (2006) pp. 65–97.

[74] M. Seel: *Implementation of planar Nef polyhedra.* Report MPI-I-2001-1-003, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, 2001.

[75] G. Taubin: "Rasterizing Algebraic Curves and Surfaces". *IEEE Computer Graphics and Applications* **14** (1994), pp. 14–23.

[76] R. J. Walker: *Algebraic Curves.* Princeton University Press, 1950.

[77] R. Wein: "High-level filtering for arrangements of conic arcs". *Proc. 10th European Symp. on Algorithms* (ESA 2002), pp. 884–895, Springer LNCS 2461, Berlin, 2002.

[78] N. Wolpert: *An Exact and Efficient Approach for Computing a Cell in an Arrangement of Quadrics.* Ph. D. thesis, Saarland University, Saarbrücken, Germany, 2002.

[79] N. Wolpert: "Jacobi Curves: Computing the Exact Topology of Arrangements of Non-Singular Algebraic Curves". *Proc. 11th European Symp. on Algorithms* (ESA 2003), pp. 532–543, Springer LNCS 2832, Berlin, 2003.

[80] C. K. Yap: *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, New York, 2000.

[81] C. K. Yap: "Robust Geometric Computation". Chapter 41 in: J. E. Goodman, J. O'Rourke (eds.): *Handbook of Discrete and Computational Geometry*, 2nd ed. CRC Press, Boca Raton, 2004.

[82] C. K. Yap: "Complete Subdivision Algorithms, I: Intersection of Bezier Curves", extended abstract.
`ftp://cs.nyu.edu/pub/local/yap/exact/bezier1.ps.gz`