3

Algebraic Issues in Computational Geometry

Bernard Mourrain*, Sylvain Pion, Susanne Schmitt, Jean-Pierre Técourt, Elias Tsigaridas, and Nicola Wolpert

3.1 Introduction

Geometric modeling plays an increasing role in fields at the frontier between computer science and mathematics. This is the case for example in CAGD (Computer-aided Geometric design, where the objects of a scene or a piece to be built are represented by parameterized curves or surfaces such as NURBS), robotics or molecular biology (rebuilding of a molecule starting from the matrix of the distances between its atoms obtained by NMR).

The representation of shapes by piecewise-algebraic functions (such as B-spline functions) provides models which are able to encode the geometry of an object in a compact way. For instance, B-spline representations are heavily used in Computed Aided Geometric Design, being now a standard for this area. Recently, we also observe a new trend involving the use of patches of implicit surfaces. This includes in particular the representation by quadrics, which are more natural objects than meshes for the representation of curved shapes.

From a practical point of view, critical operations such as computing intersection curves of parameterized surfaces are performed on these geometric models. This intersection problem, as a typical example linking together geometry, algebra and numeric computation, received a lot of attention in the past literature. See for instance [154, 275, 230]. It requires robust methods, for solving (semi)-algebraic problems. Different techniques (subdivision, lattice evaluation, marching methods) have been developed [273, 173, 14, 188, 187, 275]. A critical question is to certify or to control the topology of the result.

From a theoretical point of view, the study of algebraic surfaces is also a fascinating area where important developments of mathematics such as singularity theory interact with visualization problems and the rendering of mathematical objects. The classification of singularities [27] provides simple algebraic formulas for complicated shapes, which geometrically may be difficult to

* Chapter coordinator

handle. Such models can be visualized through techniques such as ray-tracing[1] in order to produce beautiful pictures of these singularities. Many open questions, related for instance to the topological types of real algebraic curves or surfaces, remain to be solved in this area. Computation tools, which allow to treat such algebraic models are thus important to understand their geometric properties.

In this chapter, we will describe methods for the treatment of algebraic models. We focus on the problem of computing the topology of implicit curves or surfaces. Our objective is to devise certified and output-sensitive methods, in order to combine control and efficiency. We distinguish two types of subproblems:

- the construction of new geometric objects such as points of intersection,
- predicates such as the comparison of coordinates of intersection points.

In the first case, a good approximation of the exact algebraic object, which usually cannot be described explicitly by an analytic formula, may be enough. On the contrary for the second subproblem, the result has to be exact in order to avoid incoherence problems, which might be dangerous from an implementation point of view, leading to well known non-robustness issues.

These two types of geometric problems, which appear for instance in arrangement computations (see Chapter 1) lead to the solution of algebraic questions. In particular, the construction or the comparison of coordinates of points of intersections of two curves or three surfaces involve computations with algebraic numbers. In the next section, we will describe exact methods for their treatment. Then we show how to apply these tools to compute the topology of implicit curves. This presentation includes effective aspects and pointers to software. It does not include proofs, which can be found in the cited literature.

3.2 Computers and Numbers

Geometric computation is closely tied to arithmetic, as the Ancient Greeks (in particular Pythagoras of Samos and Hippasus of Metapontum) observed a long time ago. This has been formalized more recently by Hilbert [202], who showed how geometric hypotheses are correlated with the arithmetic properties of the underlying field. For instance, it is well-known that Pappus' theorem is equivalent to the commutativity property of the underlying arithmetic field. When we want to do geometric computations on a computer, the situation becomes even more intricate. First, we cannot represent all real numbers on a computer.

Integers (even integers of unbounded size) are the basis of arithmetic on a computer. These integers are (usually) represented in the binary system as an

[1] see e.g. `http://www.algebraicsurface.net/`

array of bits; an integer n has (bit) size $O(\log|n|)$. Under this notion, integers are no longer constant size objects thus arithmetic operations on them are performed in non-constant time: for two integer of bit size $O(\log|n|)$ addition or subtraction can be done in linear time with respect to their size, i.e $O(\log|n|)$ and multiplication or division can be done in $O(\log|n|\log\log|n|\log\log\log|n|)$. Therefore, depending on the context, manipulating multi precision integers can be expensive. Dedicated libraries such as GMP[6] however have been tuned to treat such large integers.

Similarly, rational numbers can be manipulated as pairs of integer numbers. As in Pythagoras' philosophy, these numbers can be considered as the foundations of computer arithmetic. That is why, hereafter, we will consider that our input (which as we will see in the next sections corresponds to the coefficients of a polynomial equation) will be represented with rational numbers $\in \mathbb{Q}$. In other words, we will consider that the input data of our algorithms are exact. From the complexity point of view, the cost of the operations on rationals is a simple consequence of the one on integers, however we can also point out that adding rationals roughly doubles their sizes, contrary to integers, so additional care has to be taken to get good performance with rationals.

When performing geometric computations, such as for instance computing intersections, the values that we need to manipulate are no longer rationals. We are facing *Pythagoras' dilemma*: how to deal with *non-commensurable* values, when only rational arithmetic is effectively available on a computer. In our context, these non-commensurable values are defined implicitly by equations whose coefficients are rationals. As we will see, they involve algebraic numbers. A classical way to deal with numbers which are not representable in the initial arithmetic model, is to approximate them. This is usually performed by floating point numbers. For instance, numerical approximations can be sufficient, for evaluation purposes, if one controls the error of approximation. And usually, computations with approximate values is much cheaper than with the exact representation. The important problem which has to be handled is then how to control the error.

Hereafter, we describe shortly this machine floating point arithmetic and interval arithmetic, for their use in geometric computation.

3.2.1 Machine Floating Point Numbers: the IEEE 754 norm

Besides multiple-precision arithmetic provided by various software libraries, modern processors directly provide in hardware some floating point arithmetic in a way which has been standardized as the IEEE 754 norm [209]. We briefly describe the parts of this norm which are interesting in the sequel.

The IEEE 754 norm offers several possible precisions. We are going to describe the details of the so-called double precision numbers, which correspond to the `double` built-in types of the C and C++ languages. These numbers are encoded in 64 bits: 1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa.

For non-extreme values of the exponent, the real value corresponding to the encoding is simply: $(-1)^{sign} \times 1.mantissa \times 2^{exponent-1023}$. That is, there is an implicit 1 which is not represented in front of the mantissa, and the exponent value is shifted in order to be centered at zero.

Extreme values of the exponent are special: when it is zero, then the numbers are called denormalized values and the implicit 1 disappears, which leads to a nice property called gradual underflow. This property implies that there cannot be any underflow with the subtraction or the addition: $a - b = 0 \iff a = b$. The maximal value 2047 for the exponent is used to represent 4 different special values: $+\infty$, $-\infty$, qNAN, sNAN, depending on the sign bit and the value of the mantissa. Infinite values are generated by overflow situations, or when dividing by zero. A NaN (not a number) exists in two variants, quiet or signaling, and is used to represent the result of operations like $\infty - \infty$, $0 \times \infty$, $0/0$ and any operation taking a NaN as argument.

The following arithmetic operations are specified by the IEEE 754 standard: $+$, $-$, $\times$, $\div$, $\sqrt{}$. Their precise meaning depends on a rounding mode, which can have 4 values: to the nearest (with the *round-to-even* rule in case of a tie), towards zero, towards $+\infty$ and towards $-\infty$. This way, an arithmetic operation is decomposed into its exact real counterpart, and a rounding operation, which is going to choose the representable value in cases where the real exact value is not representable in the standard format. In the sequel, the arithmetic operations with directed rounding modes are going to be written as $\underline{+}$ and $\overline{\times}$, standing for addition rounded towards $+\infty$ and multiplication rounded towards $-\infty$ for example.

Finally, let us mention that the IEEE 754 norm is currently under revision, and we can expect that in the future more operations will be available in a standardized way.

3.2.2 Interval Arithmetic

Interval arithmetic is a well known technique to control accumulated rounding errors of floating point computations at run time. It is especially used in the field of interval analysis [252]. We use interval arithmetic here in the following way: we represent at run time the roundoff error associated with a variable x by two floating point numbers $\underline{x}$ and $\overline{x}$, such that the exact value of x lies in the interval $[\underline{x}, \overline{x}]$. This is denoted as the *inclusion property*.

All arithmetic operations on these intervals preserve this property. For example, the addition of x and y is performed by computing the interval $[\underline{x}\underline{+}\underline{y}, \overline{x}\overline{+}\overline{y}]$. The multiplication is slightly more complicated and is specified as

$$x \times y = [\min(\underline{x}\underline{\times}\underline{y}, \underline{x}\underline{\times}\overline{y}, \overline{x}\underline{\times}\underline{y}, \overline{x}\underline{\times}\overline{y}), \max(\underline{x}\overline{\times}\underline{y}, \underline{x}\overline{\times}\overline{y}, \overline{x}\overline{\times}\underline{y}, \overline{x}\overline{\times}\overline{y})].$$

The other basic arithmetic operations ($-, \div, \sqrt{}$) are defined on intervals in a similar way. More complex functions, like the trigonometric functions, can also be defined over intervals on mathematical grounds. However, the IEEE 754 standard does not specify their exact behavior for floating point computations, so it is harder to implement such interval functions in practice, although some libraries can help here.

Comparison functions on intervals are special, and several different semantics can be defined for them. What we are interested in here is to detect when a comparison of the exact value can be guaranteed by the intervals. Therefore looking at the intervals allows to conclude the order of the exact values in the following cases:

$$\begin{aligned} \overline{x} < \underline{y} &\Rightarrow x < y \text{ is true} \\ \underline{x} >= \overline{y} &\Rightarrow x < y \text{ is false} \\ \texttt{otherwise} &\Rightarrow x < y \text{ is unknown} \end{aligned}$$

The other comparison operators $(>, \leq, \geq, =, \neq)$ can be defined similarly.

From the implementation point of view, the difficulty lies in portability, since the IEEE 754 functions for changing the rounding modes tend to vary from system to system, and the behavior of some processors does not always match perfectly the standard. In practice, operations on intervals can be roughly 5–10 times slower than the corresponding operations on floating point numbers, this is what we observe on low degree geometric algorithms.

Interval arithmetic is very precise compared to other methods which consist in storing a central and an error values, as the IEEE 754 norm guarantees that, at each operation, the smallest interval is computed. It is possible to get more precision from it by using multiple-precision bounds, or by rewriting the expressions to improve numerical stability for some expressions [65] which improves the sharpness of the intervals.

3.2.3 Filters

Most algebraic computations are based on evaluating numerical quantities. Sometimes, like in geometric predicates, only signs of quantities are needed in the end.

Computing with multiple-precision arithmetic in order to achieve exactness is by nature costly, since arithmetic operations do not have unit cost, in contrast to floating-point computations. It is also common to observe that floating point computation almost always leads to correct results, because the error propagation is usually small enough that sign detection is exact. Wrong signs tend to happen when the polynomial value of which the sign is sought is zero, or small compared to the roundoff error propagation. Geometrically, this usually means a degenerate or nearly degenerate instance.

Arithmetic filtering techniques have been introduced in the last ten years [165] in order to take advantage of the efficiency of floating point computations, *but* by also providing a certificate allowing to determine whether the sign of the approximately computed value is the same as the exact sign.

In the case of *filter failure*, i.e., when the certificate cannot guarantee that the sign of the approximation is exact, then another method must be used to obtain the exact result: it can be a more precise filter, or it can be multiple-precision arithmetic directly.

From the complexity point of view, if the filter step succeeds often—which is expected—then the cost of the exact method will be amortized over many calls to the predicates. The probability that the filter succeeds is linked to two factors. The first is the shape of the predicate: how many arithmetic operations does it contain and how do they influence the roundoff-error (the degree of the predicate does not really matter in itself). The second factor is the distribution of the input data of the predicates, since filter failures are more common on degenerate or nearly degenerate cases.

There are various techniques which can be used to implement these filters. They vary by the cost of the computation of the certificate, and by their precision, i.e. their typical failure rate. Finding the optimal filter for a problem may not be easy, and in general, the best solution is to use a cascade of filters [70, 114]: first try the less precise and fastest one, and in case of failure, continue with a more precise and more costly one, etc. Detailed experiments illustrating this have been performed in the case of the 3D Delaunay triangulation used in surface reconstruction in [114].

We are now going to detail two important categories of filters: dynamic filters using interval arithmetic, and static filters based on static analysis of the shape of predicates.

Dynamic Filters

Interval arithmetic, as we previously described it in 3.2.2, can be used to write filters for the evaluation of signs of polynomial expressions, and even a bit more since division and square root are also defined.

Interval arithmetic is easy to use because no analysis of a particular polynomial expression is required, and it is enough to instantiate the polynomials with a given arithmetic without changing their evaluation order. It is also the most precise approach within the hardware precision since the IEEE 754 standard guarantees the smallest interval for each individual operation. We are next going to present a less precise but faster approach known as static filters.

Static Filters

Interval arithmetic computes the roundoff error at run time. Another idea which has been initially promoted by Fortune [165] is to pull more of the error computation off run time.

The basic idea is the following: if you know a bound b on the input variables $x_1, \ldots, x_n$ of the polynomial expression $P(x_1, \ldots, x_n)$, then it is possible to deduce a bound on the roundoff error ϵ_P that will occur during the evaluation of P. This can be shown inductively, by considering the roundoff error propagation bound of each operation, for example for the addition: suppose x and y are variables you want to add, b_x and b_y are bounds on $|x|$ and $|y|$ respectively, and ϵ_x and ϵ_y bounds on the roundoff errors done so far on x and y. Then it is easy to see that $|x+y|$ is bounded by $b_{x+y} = b_x + b_y$, and that the roundoff error is bounded by $\epsilon_x + \epsilon_y + b_{x+y}2^{-53}$, considering IEEE 754 double precision floating point computations. Similar bounds can be computed for subtraction and multiplication. Division does not play nicely here because the result is not bounded.

This scheme can also be refined in several directions by:

- considering independent initial bounds on the input variables,
- computing the bounds on the input and the epsilons at run time, which is usually still fast since the polynomial expressions we are dealing with tend to be homogeneous due to their geometric nature [247],
- doing some caching on this last computation [114],

Such filters are very efficient when a bound on the input is known, because the only change compared to a simple floating point evaluation is the sign comparison which is made with a constant ϵ whereas it would be with 0 otherwise. Drawbacks of these methods are that they are less precise, and so they need to be complemented by dynamic filters to be efficient in general. They are also harder to program since they are more difficult to automatize (the shape of the predicates needs to be analyzed). This is why some automatic tools have been developed to generate them from the algebraic formulas of the predicates [166, 268, 70].

3.3 Effective Real Numbers

In this section, we will consider a special type of real numbers, which we call *effective real numbers*. We will be able to manipulate them effectively in geometric computations, because the following methods are available:

- an algorithm which computes a numerical approximation of them to any precision.
- an algorithm which compares them in an exact way.

We will see that working in this sub-class of real numbers, is enough to tackle the geometric problems that we want to solve. Namely, we are interested by computing intersection points of curves, arrangements of pieces of algebraic curves and surfaces, ... This leads to the resolution of polynomial equations.

Here are some notations. A polynomial over a ring $\mathbb{L}$ of coefficients is an expression of the form

$$f(x) = a_n x^n + \cdots + a_1 x + a_0$$

where the *coefficients* $a_n \neq 0, a_{n-1}, \ldots, a_1, a_0$ are elements of $\mathbb{L}$ and the *variable* x may be regarded as a formal symbol with an indeterminate meaning. The greatest power of x appeared in f (with an non zero coefficient) is called the *degree* of f, (n in our case since $a_n \neq 0$). It is denoted $\deg(f)$. The degree of the zero polynomial is equal to $-\infty$. The coefficient a_n is called the *leading coefficient*, and denoted $\mathrm{ldcf}(f)$. The ring of polynomials with coefficient in $\mathbb{L}$, is denoted $\mathbb{L}[x]$.

We call a polynomial $g \in \mathbb{L}[x]$ a *factor* of f if there exists another polynomial $g \in \mathbb{L}[x]$ with $f = g \cdot h$. In particular, if $f = 0$, then every $g \in \mathbb{L}[x]$ is a factor of f.

In the following, we will consider polynomials with coefficient in a unitary ring $\mathbb{L}$. For instance, we can image $\mathbb{L} = \mathbb{Z}$. We denote by $\mathbb{K}$ a field containing $\mathbb{L}$. In the following, we work most of the time with $\mathbb{K}$ the field of rational numbers or its algebraic closure (that is the smallest field containing all the roots of polynomials with rational coefficients). In some cases, the problem may depend on parameters $u_1, \ldots, u_n$ and so in theses cases the field $\mathbb{K}$ will be the fraction field $\mathbb{K} = \mathbb{Q}(u_1, \ldots, u_n)$. The algebraic closure of the field $\mathbb{K}$ is denoted $\overline{\mathbb{K}}$. (so image $\overline{\mathbb{K}} = \mathbb{C}$).

3.3.1 Algebraic Numbers

We recall here the basic definitions on algebraic numbers. An *algebraic number* over the field $\mathbb{K}$ is a root of a polynomial $p(x)$ with coefficients in $\mathbb{K}$ ($p(x) \in \mathbb{K}[x]$). An *algebraic integer* over the ring $\mathbb{L}$ is a root of a polynomial with coefficients in $\mathbb{L}$, where the leading coefficient is 1.

Let α be an algebraic number over $\mathbb{K}$ and $p(x) \in \mathbb{K}[x]$ be a polynomial of degree d with $p(\alpha) = 0$. If $p(x)$ is irreducible over $\mathbb{K}$ (it cannot be written in $\mathbb{K}[x]$ as the product of two polynomials which are both different from 1), it is called the *minimal polynomial* of α. The other roots $\alpha_2, \ldots, \alpha_d$ of the minimal polynomial in $\overline{\mathbb{K}}$ are the *conjugates* of α. The *degree* of the algebraic number α is the degree of the minimal polynomial defining α. Let $\alpha_1 = \alpha$, then the *norm* of α is

$$N(\alpha) = \prod_{i=1}^{d} |\alpha_i|$$

If α, β are algebraic numbers over $\mathbb{K}$, then $\alpha \pm \beta, \alpha \cdot \beta, \alpha/\beta$ (if $\beta \neq 0$) and $\sqrt[k]{\alpha}$ are algebraic numbers over $\mathbb{K}$. If α, β are algebraic integers over $\mathbb{L}$, then $\alpha \pm \beta, \alpha \cdot \beta$ and $\sqrt[k]{\alpha}$ are algebraic integers over $\mathbb{L}$.

For instance, $\gamma = 7$ is an algebraic integer over $\mathbb{Q}$ since it is the root of $x - 7 = 0$. Moreover, $\alpha = \sqrt{2}$ (resp. $\beta = \sqrt{3}$) is an algebraic integer over $\mathbb{Q}$, since it is the positive root of the (minimal) polynomial $x^2 - 2$, (resp. $x^2 - 3$) and $\alpha + \beta$ is a root of $(x^2 - 5)^2 - 24 = x^4 - 10x^2 + 1 = 0$. We observe in the last example, that the degree of the minimal polynomial of $\alpha + \beta$ is bounded by the product of the degrees of the minimal polynomials of α and β. This is a general result, which we deduce from the resultant properties (see

Section 3.4.1 and [233]). The same result is valid for the operations $-, \times, /$ on these algebraic numbers.

Let $p(x)$ be a polynomial with algebraic number as coefficients. Then the roots of $p(x)$ are algebraic numbers. If the coefficients of $p(x)$ are algebraic integers and the leading coefficient of $p(x)$ is 1, then the roots of $p(x)$ are algebraic integers.

We describe now two important methods to represent real algebraic numbers.

3.3.2 Isolating Interval Representation of Real Algebraic Numbers

A natural way to encode a *real* algebraic number α over $\mathbb{Q}$ is by using

- a polynomial $p(x)$ of $\mathbb{Q}[x]$, which vanishes at α, and
- an *isolating interval* $[a, b]$ containing α such that $a, b \in \mathbb{Q}$ and $p(x)$ has exactly one real root in $[a, b]$.

This representation is not unique, since the size of the interval $[a, b]$ can reduce to any $\epsilon > 0$ close to 0. If we assume moreover that p is a square-free polynomial (that is $\gcd(p, p') = 1$ or in other words that the roots of p are distinct), as we assume in what follows, then α is a simple root of p and p obtains different sign when evaluated over the endpoints of the isolating interval, i.e. $p(a)p(b) < 0$.

Besides isolating interval representation there are other representations of real algebraic numbers. The most important alternative is *Thom's encoding* [42]. The basic idea behind this representation is that the signs of all the derivatives of p obtained by evaluation over the real roots of p uniquely characterize (and order) the real roots. This representation besides the uniqueness property is also more general than the isolating interval representation. However we are not going into the details.

For higher (arbitrary) degree, isolating intervals will be computed by univariate polynomial solvers, that we will describe in Section 3.4.2. For polynomials of degree up to 4, real root isolation is more effective since it can be performed in constant time (see Section 3.4.3).

3.3.3 Symbolic Representation of Real Algebraic Numbers

The sum $\alpha+\beta$ of two algebraic numbers α, β of degree $\leq d$ is an algebraic number, whose minimal polynomial is of degree $\leq d^2$ (see Section 3.4.1 and [233]). Instead of computing this minimal polynomial (which might be costly to perform), we may wonder if we can use its symbolic representation (as an arithmetic tree) and develop methods which allow us to consider it as an effective algebraic numbers. In other words,

- how can we approximate such a number within an arbitrary precision?
- how can we compare two such numbers?

In this section, we describe the symbolic representation of such algebraic numbers and in the next section, we will show how to perform these operations.

A *real algebraic expression* is an arithmetic expression built from the integers using the operations $+, -, \cdot, /, \sqrt[k]{\ }$ and $\diamond$, representing a root of a univariate polynomial, and which is defined as follows: The syntax for the $\diamond$-operator is $\diamond(j, E_d, \ldots, E_0)$, where E_i are real algebraic expressions and $1 \leq j \leq d$ is an integer. It is representing the j-th real root (if it exists) of a polynomial with coefficients $(E_i)_{i=0}^d$.

The value $\mathrm{val}(E)$ of a real algebraic expression E is the real value given by the expression (if this is defined). For $E = \diamond(j, E_d, \ldots, E_0)$, the value $\mathrm{val}(E)$ is the j-th smallest real root of the polynomial

$$p(x) = \sum_{i=0}^{d} \mathrm{val}(E_i) x^i,$$

if it exists and if the values of all coefficients are defined.

We are representing real algebraic expressions as directed acyclic graphs.The inner nodes are the operations and the leaves are integers. Every node knows an interval containing the exact value represented by its subgraph. If further accuracy is needed, the values are approximated recursively with higher precision.

Operations are done by creating a new root node and building the graph structure. Then a first approximating interval is computed. Comparisons are done exactly. The algorithms involved in these comparisons will be described in the next section.

3.4 Computing with Algebraic Numbers

In the previous section, we have described how we encode real algebraic numbers. We are going to describe now the main tools and algorithms, which allow us to compute this representation, that is

- to isolate the real roots of a polynomial.

We are also going to see how to perform the main operations we are interested in for geometric computations, namely:

- the comparison of two algebraic numbers,
- and the sign evaluation of a polynomial expression.

3.4.1 Resultant

A fundamental tool for the manipulation and construction of algebraic numbers is the resultant. It allows us to answer the following question; When do two univariate polynomials f and g of positive degree have a non-constant common factor? Since every non-contant polynomial has a root in $\overline{\mathbb{K}}$, this is equivalent to the following question: When do two univariate polynomials f and g of positive degree have a common root in $\overline{\mathbb{K}}$? Here is a first answer:

Theorem 1. *Let $f, g \in \mathbb{K}[x]$ be two polynomials of degrees $\deg(f) = n > 0$ and $\deg(g) = m > 0$. Then f and g have a non-constant common factor if and only if there exist polynomials $A \in \mathbb{K}[x]$ and $B \in \mathbb{K}[x]$ with $\deg(A) < m$ and $\deg(B) < n$ which are not both zero and such that $Af + Bg = 0$.*

Now,in order to determine the existence of a common factor of

$$\begin{aligned} f(x) &= f_n x^n + f_{n-1}x^{n-1} + \cdots + f_0, \quad f_n \neq 0,\ n > 0 \quad \text{and} \\ g(x) &= g_m x^m + g_{m-1}x^{m-1} + \cdots + g_0, \quad g_m \neq 0,\ m > 0 \end{aligned}$$

we have to decide whether two polynomials A and B with the required properties can be found. This question can be answered with the help of linear algebra: A and B are polynomials of degree at most $m-1$ and $n-1$, and therefore there are all in all $m+n$ unknown coefficients $a_{m-1}, \ldots, a_0, b_{n-1}, \ldots, b_0$ of A and B:

$$\begin{aligned} A(x) &= a_{m-1}x^{m-1} + \cdots + a_1 x + a_0 \\ B(x) &= b_{n-1}x^{n-1} + \cdots + b_1 x + b_0. \end{aligned}$$

The polynomial $A(x)f(x)+B(x)g(x)$ has degree at most $n+m-1$ in x. Each of its coefficients has to be zero in order to achieve $Af + Bg = 0$:

$$\begin{array}{ccccl} f_n a_{m-1} & + & g_m b_{n-1} & & = 0 \text{ (coefficient of } x^{m+n-1}) \\ f_{n-1}a_{m-1} + f_n a_{m-2} & + & g_{m-1}b_{n-1} + g_m b_{n-2} & & = 0 \text{ (coefficient of } x^{m+n-2}) \\ \ddots & & \ddots & & \vdots \\ & f_0 a_0 & + & g_0 b_0 & = 0 \text{ (coefficient of } x^0). \end{array}$$

We get $n+m$ linear equations in the unknowns a_i, b_i with coefficients f_i, g_j. Written in matrix style this system of linear equations has the form:

$$(a_{m-1}, \ldots, a_0, b_{n-1}, \ldots, b_0) \cdot \begin{pmatrix} f_n & f_{n-1} & \cdots & f_0 & & & \\ & f_n & f_{n-1} & \cdots & f_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & f_n & f_{n-1} & \cdots & f_0 \\ g_m & g_{m-1} & \cdots & g_0 & & & \\ & g_m & g_{m-1} & \cdots & g_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_m & g_{m-1} & \cdots & g_0 \end{pmatrix} = (0, \ldots, 0)$$

where the empty positions are filled with zeroes. We know from linear algebra that this system of linear equations has a non-zero solution if and only if the determinant of the coefficient matrix is equal to zero. This leads to the following definition:

Definition 1. *The* $(m+n)\times(m+n)$ *coefficient matrix with* m *rows of* f*-entries and* n *rows of* g*-entries*

$$\mathrm{Syl}(f,g) := \begin{pmatrix} f_n & f_{n-1} & \cdots & f_0 & & & \\ & f_n & f_{n-1} & \cdots & f_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & f_n & f_{n-1} & \cdots & f_0 \\ g_m & g_{m-1} & \cdots & g_0 & & & \\ & g_m & g_{m-1} & \cdots & g_0 & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_m & g_{m-1} & \cdots & g_0 \end{pmatrix}$$

where the empty positions are filled by zeroes is called the Sylvester matrix *of* f *and* g*. The determinant of the matrix is called the* resultant *of* f *and* g*:* $\mathrm{Res}(f,g) := \det(\mathrm{Syl}(f,g))$.

This resultant will also be denoted $\mathrm{Res}_x(f,g)$ if the coefficients a_i, b_j of f and g are themselves polynomials of other variables and we want to denote that we consider them as univariate polynomials with respect to x.

From the above observations we immediately obtain a criterion for testing whether two polynomials f and g have a non-constant common factor.

Proposition 1. *Given* $f,g \in \mathbb{K}[x]$ *of positive degree, the resultant* $\mathrm{Res}(f,g) \in k$ *is equal to zero if and only if* f *and* g *have a non-constant common factor. For* $\mathbb{K} = \mathbb{C}$ *the equality* $\mathrm{Res}(f,g) = 0$ *holds if and only if* f *and* g *have a common complex root.*

As a direct application, we see that if α (resp. β) is a root of the polynomial f (resp. g), then $\alpha+\beta$ is a root of the polynomial $\mathrm{Res}_x(f(x), g(u-x)) = R(u) = 0$ where we consider $g(u-x)$ as a polynomial in x (with coefficients which are polynomials in u). Indeed, the two polynomials $f(x)$ and $g(\alpha+\beta-x)$ have a common root $x=\alpha$, so that their resultant $R(\alpha+\beta)$ vanishes. Similarly, for $\beta \neq 0$, a defining polynomial of $\frac{\alpha}{\beta}$ is $\mathrm{Res}_x(f(x), g(x\,u)) = 0$. Though the resultant yields a direct way to compute a defining polynomial of sums, differences, products, divisions of algebraic numbers, alternative approaches, which are more interesting from a complexity point of view, have been considered (see for instance [17]). They are based on Newton sums and series expansions.

Properties of resultants.

We present some useful properties of the resultants.

Lemma 1. *Let* $f,g \in \mathbb{K}[x]$ *and* $\alpha \in \bar{\mathbb{K}}$*.*

1. *For* $\deg(f) > 0$ *and* $\deg(g) = m > 0$ *we have* $\mathrm{Res}(\alpha \cdot f, g) = \alpha^m \cdot \mathrm{Res}(f,g)$.
2. *If* $\deg(g) > 0$*, then* $\mathrm{Res}((x-\alpha)\cdot f, g) \;=\; g(\alpha)\cdot \mathrm{Res}(f,g)$.

The lemma leads to the following important characterization of resultants:

Theorem 2. *Let $f, g \in \mathbb{K}[x]$, $f_n = \mathrm{ldcf}(f)$, $g_m = \mathrm{ldcf}(g)$, $\deg(f) = n > 0$, $\deg(g) = m > 0$, with (complex) roots*

$$\alpha_1, \ldots, \alpha_n, \beta_1 \ldots, \beta_m \in \overline{\mathbb{K}}.$$

For the resultant of f and g the following holds:

$$\mathrm{Res}(f, g) \;=\; f_n^m g_m^n \prod_{i=1}^{n} \prod_{j=1}^{m} (\alpha_i - \beta_j) = f_n^m \prod_{i=1}^{m} g(\alpha_i) = (-1)^{mn} g_m^n \prod_{i=1}^{m} f(\beta_i).$$

Subresultants of Univariate Polynomials

Theorem 1 of the previous section can be generalized to

Theorem 3. *Let $f, g \in \mathbb{K}[x]$ be two polynomials of degrees $\deg(f) = n > 0$ and $\deg(g) = m > 0$. Then f and g have a common factor of degree greater than $l \geq 0$ if and only if there are polynomials A and B in $\mathbb{K}[x]$, with $\deg(A) < m - l$ and $\deg(B) < n - l$ which are not both zero, and such that $Af + Bg = 0$.*

As an immediate consequence we obtain a statement about the degree of the greatest common divisor of f and g:

Corollary 1. *The degree of the gcd of two polynomials $f, g \in \mathbb{K}[x]$ is equal to the smallest index h such that for all polynomials A and $B \in \mathbb{K}[x]$, with $\deg(A) < m - h$ and $\deg(B) < n - h$: $Af + Bg \neq 0$.*

This corollary can be reformulated in the following way:

Corollary 2. *The degree of the gcd of two polynomials $f, g \in \mathbb{K}[x]$ is equal to the smallest index h such that for all rational polynomials A and B with $\deg(A) < m - h$ and $\deg(B) < n - h$: $\deg(Af + Bg) \geq h$.*

We are interested in determining the degree of the greatest common divisor of two polynomials f and g. According to Corollary 2, we have to test in succession whether for $l = 1, 2, 3, \ldots$ there exist polynomials A and B, with the claimed restriction of the degrees such that the degree of $Af + Bg$ is strictly smaller than l. The first index h, for which this test gives a negative answer, is equal to the degree of the gcd. How can we perform such a test? We have seen in the previous section that the test for $l = 0$ can be made by testing whether the resultant of f and g is equal to zero. For $l = 1, 2, 3, \ldots$ we proceed in a similar way. Let l be a fixed index and let

$$\begin{aligned} f(x) &= f_n x^n + f_{n-1} x^{n-1} + \cdots + f_0, \quad f_n \neq 0, \quad \text{and} \\ g(x) &= g_m x^m + g_{m-1} x^{m-1} + \cdots + g_0, \quad g_m \neq 0. \end{aligned}$$

We are looking for two polynomials

$$A(x) = a_{m-l-1}x^{m-l-1} + \cdots + a_1x + a_0,$$
$$B(x) = b_{n-l-1}x^{n-l-1} + \cdots + b_1x + b_0,$$

such that $\deg(Af + Bg) < l$. There are $m + n - 2l$ unknown coefficients $a_{m-l-1}, \ldots, a_0$, $b_{n-l-1}, \ldots, b_0$. The polynomial $A(x)f(x) + B(x)g(x)$ has degree at most $n+m-l-1$. The $m+n-2l$ coefficients of $x^l, x^{l+1}, \ldots, x^{m+n-l-1}$ have to be zero in order to achieve $\deg(Af + Bg) < l$. This leads to a linear system

$$(a_{m-l-1}, \ldots, a_0, b_{n-l-1}, \ldots, b_0) \cdot S_l = (0, \ldots, 0)$$

where S_l is the submatrix of the Sylvester matrix of f and g obtained by deleting the last $2l$ columns, the last l rows of f-entries, and the last l rows of g-entries. We call $\mathrm{sr}_l(f, g) = \det S_l$ the l^{th} *subresultant* of f and g. For $l = 0$, the equality $\mathrm{Res}(f, g) = \mathrm{sr}_0(f, g)$ holds. In fact, S_l is a submatrix of S_i for $l > i \geq 0$. The $2\,l \times 2\,l$ minors of the submatrix of the Sylvester matrix of f and g obtained by deleting the last l rows of f-entries, can be collected in order to construct a polynomial, which has interesting properties. To be more specific, we need the following definition:

Definition 2 (Determinant polynomial).
Let $\mathcal{M}$ be a $s \times t$ matrix, $s \leq t$, over an integral domain $\mathbb{L}$. The determinant polynomial of $\mathcal{M}$ is:

$$\mathrm{detpol}(\mathcal{M}) = |\mathcal{M}_s|\, x^{t-s} + \cdots + |\mathcal{M}_t|$$

where $\mathcal{M}_j$ denotes the submatrix of $\mathcal{M}$ consisting of the first $s-1$ columns followed by the j^{th} column, for $s \leq j \leq t$.

Definition 3 (Subresultant).
Let $f, g \in \mathbb{K}[x]$, two polynomials with $deg(f) = n > 0$, $deg(g) = m > 0$. For $0 \leq l \leq \min(f, g)$, we define:

$$M_l = \mathrm{mat}(x^{n-l-1}f(x), x^{n-l-2}f(x), \ldots, f(x), x^{m-l-1}g(x), \ldots, g(x))$$

Then the l^{th} subresultant polynomial of f and g is $\mathrm{Sr}_l(f, g)(x) = \mathrm{detpol}(M_l)$.

Notice that the coefficient of x^l in $\mathrm{Sr}_l(f, g)$ is the l^{th} subresultant coefficient, denoted $\mathrm{sr}_l(f, g)$. Here is the main proposition [42, 326]:

Proposition 2. *Two polynomials f and g of positive degree have a gcd of degree h if and only if h is the least index l for which $\mathrm{sr}_l(f, g) \neq 0$. In this case, their gcd is $\mathrm{Sr}_l(f, g)(x)$.*

This yields Algorithm 6 for computing the square-free part of a polynomial. By Hadamard's identity (see [42]), the size of the coefficients of the subresultants is bounded linearly (up to a logarithmic factor) in terms of the size of the minors.

Algorithm 6 Square free part of a univariate polynomial

INPUT: a polynomial $f \in \mathbb{K}[x]$.

- Compute the last non-zero subresultant $\mathrm{Sr}(x)$ polynomial of $f(x)$ and $f'(x)$.
- Compute $f^r = f/\mathrm{Sr}(x)$.

OUTPUT: the square-free part f^r of f.

3.4.2 Isolation

We are now going to describe algorithms that compute isolating intervals for the real roots of polynomials of arbitrary degree with rational coefficients. In the next section we will present how isolating intervals can be computed directly for polynomials of degree up to 4. Two families of methods will be considered, one based on Descartes's rule and the other on Sturm sequences. Both produce certified isolation intervals for all the roots of a polynomial and have interesting complexity bounds.

Descartes' rule.

Consider a polynomial $p(x) \in \mathbb{R}[x]$, represented in the monomial basis by

$$p(x) = a_d\, x^d + \cdots + a_0,$$

with $a_d \neq 0$. A simple rule to estimate the number of real roots of p is obtained by counting the number of sign changes of its coefficients:

Definition 4 (Sign changes). *The number of sign changes $V(\mathbf{a})$ in a sequence $\mathbf{a} = \{a_0, \ldots, a_d\}$ in $\mathbb{R} - \{0\}$ is defined, by induction on d, by:*

- $V(a_0) = 0$,
- $V(a_0, \ldots, a_d) = V(a_1, \ldots, a_d) + 1$ *if* $a_0 a_1 < 0$,
- $V(a_0, \ldots, a_d) = V(a_1, \ldots, a_d)$ *if* $a_0 a_1 > 0$.

This definition extends to any finite sequence a of elements in $\mathbb{R}$ by considering the finite sequence b obtained by dropping the zeroes in a and defining $V(a) = V(b)$, with the convention $V(\emptyset) = 0$.

Here is a simple but interesting result, known as Descartes' lemma [111]:

Lemma 2 (Descartes). *Let $p(x) = a_d\, x^d + \cdots + a_0$, let $V(\mathbf{a})$ be the number of sign changes in the sequence $\mathbf{a} = \{a_0, a_1, \ldots, a_d\}$ and let $N_+(p)$ be the number of strictly positive roots of p (counted with multiplicity). Then*

$$N_+(p) = V(\mathbf{a}) - 2\,k,$$

with $k \in \mathbb{N}$.

This rule can be used to isolate the roots of a polynomial p, as shown hereafter.

Let us recall that a univariate polynomial $p(x)$ of degree d can also be represented as:

$$p(x) = \sum_{i=0}^{d} b_i \, B_d^i(x),$$

where $B_d^i(x) = \binom{d}{i} x^i (1-x)^{d-i}$. The polynomials $(B_d^i)_{i=0,\ldots,d}$ form the so-called Bernstein basis in degree d on $[0,1]$. The sequence $\mathbf{b} = [b_i]_{i=0,\ldots,d}$ is called the set of control coefficients on $[0,1]$. Similarly, we will say that a sequence $\mathbf{b}$ represents the polynomial function p on the interval $[a,b]$ if:

$$p(x) = \sum_{i=0}^{d} b_i \, \binom{d}{i} \frac{1}{(b-a)^d} \, (x-a)^i (b-x)^{d-i}.$$

The polynomials

$$B_d^i(x; a, b) := \binom{d}{i} \frac{1}{(b-a)^d} \, (x-a)^i (b-x)^{d-i}$$

form the Bernstein basis in degree d on $[a,b]$. We are going to consider the sequence of values $\mathbf{b}$ together with the corresponding interval $[a,b]$. A first property of this representation is that the derivative f' of f, is deduced easily from the control coefficients:

$$d\Delta\mathbf{b} := d(b_{i+1} - b_i)_{0 \leqslant i \leqslant d-1},$$

where $\Delta\mathbf{b} = (\Delta b_0, \ldots, \Delta b_{d-1})$ with $\Delta b_i = b_{i+1} - b_i$. Another fundamental algorithm that we will use is the de Casteljau algorithm [153]:

$$\begin{aligned} b_i^0 &= b_i \quad i = 0, \ldots, d \\ b_i^r &= (1-t)\, b_i^{r-1} + t\, b_{i+1}^{r-1}(t) \quad i = 0, \ldots, d-r \end{aligned}$$

It allows us to compute the representation of p on the subdivided intervals $[a, (1-t)a + tb]$ and $[(1-t)a + tb, b]$. For a complete list of methods on this representation, we refer to [153].

By substituting x by $x = \frac{t}{1-t}$, the Descartes rule yields the following proposition [153], [290]:

Proposition 3. *The number of sign changes $V(\mathbf{b})$ of the control coefficients $\mathbf{b} = [b_i]_{i=0,\ldots,d}$ of a univariate polynomial on $[0,1]$ bounds its number of real roots in $[0,1]$ and is equal to it modulo 2. Thus,*

- *if $V(\mathbf{b}) = 0$, there is no real root in $[0,1]$, and*
- *if $V(\mathbf{b}) = 1$, there is exactly one real root in $[0,1]$.*

Sturm sequences.

Another tool which can also be used for the isolation of real roots is described below.

Definition 5 (Sturm sequence). *Let p, q be two univariate polynomials. A polynomial sequence $f_0 = p, f_1 = q, \dots, f_s$ is a Sturm sequence if:*

- *f_s divides all the f_i, $0 \le i \le s$. Let $\delta_i = f_i/f_s$, $0 \le i \le s$.*
- *If c is a real number such that $\delta_j(c) = 0$, $0 < j < s$, then $\delta_{j-1}(c)\delta_{j+1}(c) < 0$.*
- *If c is a real number such that $\delta_0(c) = 0$ then $\delta_0(x)\delta_1(x)$ has the sign of $x - c$ in a neighborhood of c.*

We will denote the Sturm sequence by $Sturm(p, q) = (p, q, f_2, \dots, f_s)$. Such a Sturm sequence can be computed for instance, as follows:

$$f_0 = p, f_1 = q, f_2 = -\mathrm{rem}(f_0, f_1), \dots f_{i+1} = -\mathrm{rem}(f_{i-1}, f_i), \dots$$

where $\mathrm{rem}(f, g)$ is the reminder in the Euclidean division of the polynomial f by the polynomial g.

For any sequence S of real polynomials and $a \in \mathbb{R}$, we denote by $V(S, a)$ the number of sign variations of the values of the polynomials in S at a. Then, we have the well-known theorem of Sturm (see for instance [42]):

Proposition 4 (Sturm theorem). *Assume $S = Sturm(p, p'q)$ and $]a, b[$ is an interval such that $p(a)p(b) \neq 0$. The difference $V(S, a) - V(S, b)$ is equal to the difference between the number $Z_p(q > 0; a, b)$ of roots α of p in $]a, b[$ (without multiplicity) such that $q(\alpha) > 0$ and the number $Z_p(q < 0; a, b)$ of roots α of p in $]a, b[$ such that $q(\alpha) < 0$:*

$$V(S, a) - V(S, b) = Z_p(p; a, b) - Z_p(q < 0; a, b).$$

If we take $q = 1$, then we have that $V(S, a) - V(S, b)$ is the number of real roots of p in $]a, b[$.

An efficient way to compute a Sturm sequence is to compute a Sturm-Habicht (or Sylvester-Habicht) sequence. For that purpose, we recall here the definition of pseudo-remainder (see for instance [42]).

Definition 6 (Pseudo-remainder). *Let*

$$f(x) = a_p x^p + \dots + a_0$$

$$g(x) = b_q x^q + \dots + b_0$$

be two polynomials in $\mathbb{L}[x]$ where $\mathbb{L}$ is a ring. Note that the only denominators occurring in the euclidean division of f by g are $b_q^i, i \le p + q - 1$.
The signed pseudo-remainder denoted $\mathrm{prem}(f, g)$*, is the remainder in the Euclidean division of $b_q^d f$ by g, where d is the smallest even integer greater than or equal $p - q + 1$. Note that the euclidean division of $b_q^d f$ by g can be performed in $\mathbb{L}$ and that* $\mathrm{prem}(f, g) \in \mathbb{L}[x]$.

Definition 7 (Sturm-Habicht sequence). *Let p and q be univariate polynomials, $d = \max(\deg(p), \deg(q) + 1)$, $\mathrm{coef}_k(p)$ the coefficient of x^k in p, and $\delta_k = (-1)^{k(k-1)/2}$.*

The Sturm-Habicht sequence of p and q is defined inductively as follows:

- $H_d = p$, $h_d = 1$.
- $H_{d-1} = q$.

Assume that we have computed $H_d, \ldots, H_{j-1}$, $h_d, \ldots, h_j$ with $h_j \neq 0$ and $H_{j-1} \neq 0$. Let $k = deg(H_{j-1})$. Then:

- *If $k < j-1$, let $H_k = \delta_{j-k} \frac{\mathrm{coef}_k(H_{j-1})^{j-1-k}}{h_j^{j-1-k}} H_{j-1}$, $h_{j-1} = 1$.*
  For $l \in \mathbb{N}$ with $k < l < j-1$, let $H_l = 0$, $h_l = 0$.
- *Let $h_k = \mathrm{coef}_k(H_k)$, $H_{k-1} = \delta_{j-k+2} \frac{\mathrm{prem}(H_j, H_{j-1})}{h_j^{j-k+1}}$ (where* prem *is the pseudo-remainder)*

The main interest of this construction is that the polynomials in the sequence are related to the subresultants of p and q (see Section 3.4.1). Therefore, if we are working with coefficients in a ring $\mathbb{L}$, the coefficients of these subresultant polynomials are in the ring $\mathbb{L}$. If for instance $\mathbb{L} = \mathbb{K}[u]$, the subresultant polynomials are polynomials in x, which coefficients are polynomials in the variable u. By substituting u by a value $u_0 \in \mathbb{K}$, we get by construction, the subresultant sequence, evaluated at $u = u_0$. In other words, subresultant sequences behave well under specialization.

This Sturm-(Habicht) sequence can also be useful for gcd computations, since the gcd corresponds to the last non-zero term of the sequence. In particular, it yields anotehr way to compute the square-free part $p/\gcd(p, p')$ of a polynomial $p \in \mathbb{Q}[x]$ (see Algorithm 6).

Isolation algorithm.

The idea behind both approaches that we are goping to present now, is to consider an interval that initialy contains all the real roots, and then repeatedly subdivide it until we obtain intervals that is guaranteed that they contain zero or one real root.

Both Descartes's and Sturm's approach gives a bound on the number of roots of a polynomial f on an interval I, that we denote hereafter by $V(f; I)$. For Descartes' approach, $V(f; I)$ is the number of sign changes in the coefficients of f in the Bernstein basis on I. This bound is equal to the number of roots modulo 2. For the Sturm approach, $V(f; I)$ is the difference of the number of sign changes of the Sturm sequence of f and f', evaluated over the end points of the interval. Notice that in this approach we count exactly the number of real roots of f in the interval I.

This leads to the algorithm 7.

In the Sturm approach, we compute first the Sturm sequence of f and f'. Then, in order to compute $V(f; I)$, we evaluate the sign of the polynomials of this Sturm sequence at the end points of the interval I. If we use

Algorithm 7 Isolation of real roots

INPUT: A polynomial $f \in \mathbb{Z}[x]$.

- Compute the square-free part f^r of f (Algorithm 6).
- Compute an interval I_0 containing the roots of f and initialize a queue Q with I_0.
- While Q is not empty,
  - Pop an interval I from Q and compute $V(f, I)$.
  - If $V(f; I) > 1$, split the interval in two subintervals I_- and I_+ and add them to Q.
  - If $V(f; I) = 0$, then remove the interval I.
  - If $V(f; I) = 1$, then output I.

OUTPUT: a list of intervals, with as many real roots of f than intervals and such that each interval contains exactly one real root of f.

the Sturm-Habicht construction, as $f \in \mathbb{Z}[x]$ the polynomials of the Sturm-Habicht sequence will also be in $\mathbb{Z}[x]$. Moreover, if the bit size of the coefficients of the polynomial f of degree d is bounded by τ, then the bit size of the coefficients of the polynomials in the Sturm sequence are bounded by $\mathcal{O}(d\,\tau)$ (using Hadamard inequality). This algorithm can be refined by restricting the computation to the roots in a given interval.

For the Descartes' approach, we represent the polynomial f by an interval I and its coefficients in the Bernstein basis on I. Splitting the interval I consists in applying the de Casteljau algorithm in order to get the representation of f on the two subintervals I_-, I_+. Another variant consists in avoiding the square-free computation and stopping the subdivision if an interval is of size smaller than a given ϵ. This does not guarantees the result in the presence of a cluster of roots, but only bounds the multiplicity of this cluster.

A first version of the Bernstein approach was described in [232]. Its complexity was first analyzed in [263], and then improved in [257]. The proof of termination of the algorithm is based on a partial inverse of Descartes' rule, namely the two circle theorem [98, 257], which, roughly speaking, guarantees that the algorithm, after a number of steps, will produce polynomials with zero or one sign variation. Using a refined version of this theorem, and Davenport-Mahler bounds on the product of distances between pairs of (complex) roots of f (see [229]), one can even show that the number of subdivisions (for both approaches) is bounded by $\mathcal{O}(d\,\tau + d \log d)$, where d is the degree of f and τ a bound on the size of its coefficients (see also [308, 143]).

The correspondence with another algorithm based on Descartes' rule in the monomial basis, known as *modified Uspensky* algorithm [294], has been detailed in [257]. It is shown in particular, that de Casteljau algorithm can be reinterpreted as a shift of a univariate polynomial. This yields a bound $\mathcal{O}(d^4\tau^2)$ on the bit complexity for the isolation algorithm, which actually holds for both approaches. In practice, Descartes' approach is usually more efficient than the Sturm-Sequence approach (see [143] for experimentations).

Notice that Descartes' approach (with polynomials represented either in monomial or Bernstein basis) can be applied with interval arithmetic, by adapting the sign variation count [294, 257]. It also extends naturally to B-splines which are piecewise polynomial functions [153].

3.4.3 Algebraic Numbers of Small Degree

It is known that solving a polynomial equation by radicals is always possible for small degree ($d \leq 4$). However when we want to compute the real roots then we must do computations with complex numbers (for degrees 3 and 4). Thus even when explicit formulas are available it is not always possible or effective to do computations with them. Thus, we are not interested by describing the roots as explicit formulas, involving radicals for instance (since tis is not always possible).

Instead, our goal is to describe the roots of a polynomial by using only rational numbers. We will see that for small degree polynomials, of degree up to 4, we can compute the isolating interval representation of the real roots with minimal computations, while for arbitrary degree polynomials this task requires a more involved machinery (as presented in the previous section).

The interest of such approach has already been illustrated by intensive computations with algebraic numbers of degree 2. In this case, classical explicit formulas (involving simple square roots) allow us to represent the real roots and compute with them directly. Paradoxically, as it is shown in [113], it can be more interesting to use instead isolating interval representations, from an efficiency point of view. Such an approach has been recently extended for polynomials up to degree 4, in [145]. It yields a way to precompute isolating intervals offline, which proceeds as follows (see [145, 144] for more details; see also [216]).

Since algebraic numbers of degree up to 4 appear in many problems of computational geometry, for example arrangement of conic arcs in the plane, Voronoi diagrams of circles, etc, we present now how to compute the isolating interval representation of these numbers.

Given a polynomial $p \in \mathbb{Q}[x]$ of degree 4

$$p(x) = a\,x^4 - 4\,b\,x^3 + 6\,c\,x^2 - 4\,d\,x + e$$

with $a > 0$, we define the invariants

$$\begin{array}{ll}
A \;= ae - 4bd + 3c^2, & B \;= ace + 2bcd - ad^2 - eb^2 - c^3, \\
\Delta_1 = A^3 - 27B^2, & \Delta_2 = b^2 - ac, \\
\Delta_3 = c^2 - bd, & \Delta_4 = d^2 - ce, \\
W_1 = ad - bc, & W_2 = be - cd, \\
W_3 = ae - bd, & R \;= -2b^3 - a^2d + 3abc, \\
T \;= -9W_1^2 + 27\Delta_2\Delta_3 - 3W_3\Delta_2 &
\end{array}$$

The following proposition gives the number of real roots of a quartic polynomial.

Proposition 5. *Let $p(x)$ be the quartic as above. The following table gives the numbers of real roots and their multiplicities in all cases. For example, $\{1,1,1,1\}$ means four simple real roots and $\{2,2\}$ means two double real roots.*

condition	*real roots*
$\Delta_1 > 0 \wedge T > 0 \wedge \Delta_2 > 0$	$\{1,1,1,1\}$
$\Delta_1 > 0 \wedge (T \leq 0 \vee \Delta_2 \leq 0)$	$\{\}$
$\Delta_1 < 0$	$\{1,1\}$
$\Delta_1 = 0 \wedge T > 0$	$\{2,1,1\}$
$\Delta_1 = 0 \wedge T < 0$	$\{2\}$
$\Delta_1 = 0 \wedge T = 0 \wedge \Delta_2 > 0 \wedge R = 0$	$\{2,2\}$
$\Delta_1 = 0 \wedge T = 0 \wedge \Delta_2 > 0 \wedge R \neq 0$	$\{3,1\}$
$\Delta_1 = 0 \wedge T = 0 \wedge \Delta_2 < 0$	$\{\}$
$\Delta_1 = 0 \wedge T = 0 \wedge \Delta_2 = 0$	$\{4\}$

The next theorem provides a tool to isolate these real roots.

Theorem 4. *Given a polynomial $p(x)$ with two successive real roots γ_1 and γ_2, and given two other polynomials $B(x)$ and $C(x)$, define*

$$A(x) := B(x)p'(x) + C(x)p(x).$$

Then $A(x)$ or $B(x)$ has at least one real root in the closed interval $[\gamma_1, \gamma_2]$.

The idea developed in [144] is to determine the number of real roots of the polynomial $p(x)$ using the invariants and then to find isolating numbers by providing clever choices for the polynomials $A(x)$ and $B(x)$, i.e the real roots of $A(x)$ and $B(x)$ are computed easily and ideally are rationals. of Theorem 4. In the case of two simple or four simple real roots, the isolating point that we get from the initial polynomial may not be rational. In [144] the authors show how to get a rational isolating point in these cases. The following polynomials are suggested:

$$\begin{aligned} A(x) &= 3\Delta_2 x^2 + 3W_1 x - W_3 \\ B(x) &= ax - b \\ C(x) &= -4a. \end{aligned}$$

The isolating points are then in the set of real roots of $A(x)$ and $B(x)$. As a second possibility, they suggest to take the following three polynomials

$$\begin{aligned} A(x) &= W_3 x^3 - 3W_2 x^2 - 3\Delta_4 x \\ B(x) &= dx - e \\ C(x) &= -4d \end{aligned}$$

Note that the real roots of $A(x)$ might be real algebraic numbers of degree 2 in both cases, thus not rational numbers in general. However it was proven

[144] that using the ceiling of the real roots of $A(x)$ suffices in all the cases to compute isolating intervals for real roots of $p(x)$. The case of the cubic is even more straightforward, since only one choice of polynomials $A(x)$ and $B(x)$ is needed (see [144] for more details).

3.4.4 Comparison

An important operation which needs to be performed exactly on real algebraic numbers in geometric algorithms is the comparison. We describe here methods to achieve this goal.

Comparison of isolated interval algebraic numbers.

Let us describe briefly how we use Sturm's theorem to compare two algebraic numbers $\alpha = (p,]a, b[)$ and $\beta = (q,]c, d[)$, assuming for simplicity that α and β are *simple roots* of p and q. If $b < c$ (resp. $d < a$) we have $\alpha < \beta$ (resp. $\beta < \alpha$). Let us assume now that $a < c < b < d$ (the other cases being treated similarly). First we compute the sign s of $p(a)p(c)$. If $s < 0$, then we have $\alpha \in]a, c[$ and $\alpha < \beta$. If $s = 0$, we have $\alpha = c$ (since $\alpha \neq a$), which implies that $\alpha < \beta$. Otherwise $s > 0$, p has no root in the interval $[a, c]$. We compute $S = Sturm(p, p'q)$ and $v := V(S, c) - V(S, b)$. Let us assume first that $q(c) > 0$, $q(b) < 0$. Then if $v = 1$, by Sturm's theorem $q(\alpha) > 0$ and $\alpha < \beta$. If $v = -1$, $q(\alpha) < 0$ and $\alpha > \beta$. If $v = 0$, then $q(\alpha) = 0$ and $\alpha = \beta$. If now $q(c) < 0$, $q(b) > 0$, we negate the previous output. Finally, if $q(c)$ and $q(b)$ are of the same sign, then $\alpha < \beta$.

Regarding the complexity of this method, the effective computation of these sequences can be done using Sturm-Habicht sequences. For two polynomials p and q of degree bounded by d and coefficient bit size bounded by τ the bit size of the coefficients in the Sturm-Habicht sequence is $\mathcal{O}(d\tau)$ and the computation of the sequence is made with $\mathcal{O}(d^2)$ arithmetic operations [295, 42]. The reader may refer to [143], for a complete complexity analysis.

For algebraic numbers of small degree, these operations can even be precomputed, as described in [144], in order to accelerate the computation of the Sturm sequences of specific polynomials, taking advantage of the good specialization properties of the Sturm-Habicht sequences. Moreover, since the isolating interval representation of algebraic numbers of degree up to 4 can be computed in constant time we conclude that comparison of such numbers can also be performed in constant time.

For example [144], the Sturm sequence of a quartic $p(x)$ and its derivative $p'(x)$ is

$$
\begin{aligned}
S_0(x) &= p(x) \\
S_1(x) &= p'(x) \\
S_2(x) &= 3\Delta_2 x^2 + 3W_1 x - W_3 \\
S_3(x) &= T_1 x + T_2 \\
S_4(x) &= -\Delta_1
\end{aligned}
$$

where $T_1 = -W_3\Delta_2 - 3W_1^2 + 9\Delta_2\Delta_3$ and $T_2 = AW_1 - 9bB$. The Sturm sequences of two general quartic polynomials can be described similarly [144]. The precomputation of these Sturm sequences for polynomials of degree ≤ 4 reduces many of the algorithmic problems on small degree algebraic numbers to the computation of signs of algebraic expressions. The degree of these algebraic expressions is an indicator of the complexity of the approach [144]:

Theorem 5. *There is an algorithm that compares any two roots of two quartics using Sturm sequences and isolating intervals from above where the algebraic degree of the quantities involved is in the worst case at most* 14.

Notice that the comparison of two algebraic numbers can also be performed, using the Bernstein-basis approach. Indeed, suppose that the two algebraic numbers α and β are defined by the polynomials p and q. Then by computing their gcd and dividing by it, we may assume that p and q are co-prime. In this case, we can subdivide the isolation intervals of α and β using Algorithm 7, until say, q has a constant sign on the interval I_α defining α. Then I_α can be set-subtracted from the interval I_β defining β and the comparison of α and β reduces to the easy comparison of the non-overlapping intervals I_α and I_β.

Comparison of symbolic expressions.

Algorithms for the comparison of symbolic expressions representing algebraic numbers can also used. Let E be a real algebraic expression given as directed acyclic graph (dag) (source nodes are operands and internal nodes are operators). A *separation bound* $\mathrm{sep}(E)$ is a real number $\mathrm{sep}(E) > 0$, that can be computed easily from E, such that

$$\mathrm{val}(E) = 0 \quad \text{or} \quad |\,\mathrm{val}(E)| \geq \mathrm{sep}(E).$$

Separation bounds allow us to determine the sign of an expression by means of numerical computations. An error bound Δ is initialized to some positive value, say $\Delta = 1$, and an approximation $\widetilde{\xi}$ of $\xi = \mathrm{val}(E)$ with $|\xi - \widetilde{\xi}| \leq \Delta$ is computed using approximate arithmetic, say floating point arithmetic with arbitrary-length mantissa. If $|\widetilde{\xi}| > \Delta$, the sign of ξ is equal to the sign of $\widetilde{\xi}$. Otherwise, $|\widetilde{\xi}| \leq \Delta$ and hence $|\xi| < 2\Delta$. If $2\Delta \leq \mathrm{sep}(E)$, we have $\xi = 0$. If $2\Delta > \mathrm{sep}(E)$, we halve Δ and repeat the process. The worst case complexity of the procedure just outlined is determined by the separation bound.

The following separation bound are taken from the article [69]. We need to define the weight $D(E)$ of an expression, which is an upper bound on the algebraic degree of the algebraic number defined by the expression. The *weight* $D(E)$ of an expression dag E is the product of the weights of the nodes and leaves of the dag. Leaves and $+$, $-$, $\cdot$ and $/$ operations have weight 1, a $\sqrt[k]{\ }$-node has weight k, and a $\diamond(j, E_d, \ldots)$-operation has weight d.

Let $p(x) = x^d + a_{d-1}x^{d-1} + \ldots + a_0$ be a polynomial with real coefficients a_i. A *root bound* $\Phi(a_{d-1}, \ldots, a_0)$ is an upper bound for the absolute value of all roots of $p(x)$. One example for a root bound is the Lagrange-Zassenhaus bound (see [340])

$$\Phi(\ldots, a_i, \ldots) = 2 \max_{i=0,\ldots,d-1} \left\{ \sqrt[d-i]{|a_i|} \right\}.$$

Theorem 6. *Let E be a real algebraic expression with* $\mathrm{val}(E) = \xi$. *Let $u(E)$ and $l(E)$ be defined inductively on the structure of E according to the following rules:*

	$u(E)$	$l(E)$
integer N	$\lvert N\rvert$	1
$E_1 \pm E_2$	$u(E_1) \cdot l(E_2) + l(E_1) \cdot u(E_2)$	$l(E_1) \cdot l(E_2)$
$E_1 \cdot E_2$	$u(E_1) \cdot u(E_2)$	$l(E_1) \cdot l(E_2)$
E_1/E_2	$u(E_1) \cdot l(E_2)$	$l(E_1) \cdot u(E_2)$
$\sqrt[k]{E_1}$ *and* $u(E_1) = 0$	0	1
$\sqrt[k]{E_1}$ *and* $u(E_1) \geq l(E_1)$	$\sqrt[k]{u(E_1) l(E_1)^{k-1}}$	$l(E_1)$
$\sqrt[k]{E_1}$ *and* $u(E_1) < l(E_1)$	$u(E_1)$	$\sqrt[k]{(u(E_1)^{k-1} l(E_1)}$
$\diamond(j, E_d, \ldots, E_0)$	$\Phi(\ldots, \left(l(E)^{d-i-1} u(E_i) \prod_{k \neq i} l(E_k)\right), \ldots)$	$u(E_d) \prod_{k \neq d} l(E_k)$

Let $D(E)$ be the weight of E. Then either $\xi = 0$ or

$$\left(l(E) u(E)^{D(E)-1}\right)^{-1} \leq |\xi| \leq u(E) l(E)^{D(E)-1}$$

Separation bounds have been studied extensively in computer algebra [73, 249, 340, 299] as well as in computational geometry [71, 338, 69, 239].

The separation bounds given above can be improved using the fact that powers of integers can be factored out from the number *before* computing the separation bound [280, 301].

3.5 Multivariate Problems

Geometric problems in 3D often involve the solution of polynomial equations in several variables. This problem can be reduced to a univariate problem and thus to the manipulation of real algebraic numbers, as follows. We are interested in the case of zero-dimensional systems, i.e. systems that have only finitely many complex solutions.

We denote by $f_1 = 0, \ldots, f_m = 0$ the polynomial equations in $\mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1, \ldots, x_n]$ that we want to solve. The quotient ring $\mathbb{K}[\mathbf{x}]/(f_1, \ldots, f_m)$ of polynomials modulo $f_1, \ldots, f_m \in \mathbb{K}[x_1, \ldots, x_n]$ is denoted by $\mathcal{A}$. In the case that we consider here, where the number of complex roots is finite, the quotient algebra $\mathcal{A}$ is a finite-dimensional vector space over $\mathbb{K}$.

We consider the operators of multiplication M_g by an element g in the ring $\mathcal{A}$:

$$\begin{aligned} M_g : \mathcal{A} &\to \mathcal{A} \\ a &\mapsto g\,a \end{aligned}$$

Then, the algebraic solution of the system is performed by analyzing the eigenvalues and eigenvectors of these operators. It is based on the next theorem which involves the transposed of the operators M_g^t. By definition, M_g^t is acting on the set of linear forms $\widehat{\mathcal{A}}$ on $\mathcal{A}$. If ζ is a root of the polynomial system $f_1 = 0, \ldots, f_m = 0$, then the map $\mathbf{1}_\zeta : g \mapsto g(\zeta)$ is an element $\widehat{\mathcal{A}}$.

Here is the main theorem from which we deduce the root computation:

Theorem 7. *Assume that the set of complex solutions of $f_1 = 0, \ldots, f_s = 0$ is the finite set $\{\zeta_1, \ldots, \zeta_d\}$.*

1. *Let $g \in \mathcal{A}$. The eigenvalues of the operator M_g (and its transpose M_g^t) are $g(\zeta_1), \ldots, g(\zeta_d)$.*
2. *The common eigenvectors of (M_g^t) for $g \in \mathcal{A}$ are (up to a scalar) the evaluation $\mathbf{1}_{\zeta_1}, \ldots, \mathbf{1}_{\zeta_d}$.*

The first point of this theorem can be found in [37] and the second in [256]. This theorems implies in particular that the common eigenvectors of the transpose of the operators $M_{x_1}, \ldots, M_{x_n}$ of multiplication by $x_1, \ldots, x_n$, correspond to the linear forms which evaluate a polynomial at a root. A numerical approximation of these roots of the system can thus be obtained by computing the common eigenvectors of these operators, using standard linear algebra tools. See [37, 101, 256, 261, 322, 104, 139, 262] for more details on this approach.

These operators can also be used to describe the solution points as the image, by a rational map, of the roots of a univariate polynomial. In other words, the (real) coordinates of the solutions are rational functions evaluated at real algebraic numbers whose defining equations can be deduced explicitly from the matrices M_{x_i}. It leads to Algorithm 8, which yields the so-called *Rational Univariate Representation* (RUR) of the roots. For details on this

Algorithm 8 Rational Univariate Representation of the roots

INPUT: The tables $M_{x_1}, \ldots, M_{x_n}$ of multiplication by $x_1, \ldots, x_n$ in $\mathcal{A}$.

1. Compute the determinant $\Delta(\mathbf{u}) := \det(u_0\,\mathrm{I} + u_1\,M_{x_1} + \cdots + u_n\,M_{x_n})$ and its square-free part $d(\mathbf{u})$.
2. Choose a generic $t \in \mathbb{K}^{n+1}$ and compute the $n+1$ first coefficients of

$$d(t + \mathbf{u}) = d_0(u_0) + u_1\,d_1(u_0) + \cdots + u_n\,d_n(u_0) + \cdots$$

   considered as a polynomial in $u_1, \ldots, u_n$.

OUTPUT: the roots of the system $f_1 = 0, \ldots, f_m = 0$ are

$$\zeta_1 = \frac{d_1(\alpha)}{d_0'(\alpha)}, \ldots, \zeta_n = \frac{d_n(\alpha)}{d_0'(\alpha)}$$

for all roots α of the univariate polynomial equation $d_0(\alpha) = 0$.

construction, see [242, 183, 293, 42, 138].

The generic condition required in Algorithm 8 on $t \in \mathbb{K}^{n+1}$ is that it *separates* the roots: $\sum_{i=0}^{n} \zeta_i t_i \neq \sum_{i=0}^{n} \zeta'_i t_i$ if ζ and ζ' are two distinct solutions of the system. Methods to find a generic t are described for instance in [293].

In order to get a minimal rational univariate representation, one can factorize $d_0(u_0)$ and keep the irreducible factors, which divide the numerator of the fraction obtained by substituting x_i with $\frac{d_i(u_0)}{d'_0(u_0)}$.

This Rational Univariate Representation (RUR) allows us to replace the treatment of solutions of a multivariate system by the manipulation of algebraic numbers of degree bounded above by the number of complex solutions of the system.

Another important aspect is that we can compute a RUR of a polynomial system with coefficients in an algebraic extension $\mathbb{Q}[\theta]$, for θ an algebraic number, i.e with coefficients in $\mathbb{Q}[x]/(p(x))$, where p is the minimal polynomial of θ. Computing the tables of multiplication and the RUR of the roots over $\mathbb{Q}[\theta]$ require field arithmetic operations and equality test in $\mathbb{Q}[\theta]$, which are performed easily by reduction modulo p.

3.6 Topology of Planar Implicit Curves

As an application of these algebraic techniques, we detail the computation of the topology of an implicit curve. It is a key ingredient of many geometric problems including arrangement computation on arcs of curves, intersection of surfaces, ... We consider first a curve $\mathcal{C}$ defined as the zero locus $\mathcal{V}(f)$ of a polynomial in two variables $f(x, y) \in \mathbb{Q}[x, y]$. We can assume that f is square-free (otherwise, we perform Algorithm 6). In Section 3.6.1, we are going to present from a geometric point of view the way the topology is computed. In this computation, we need to manipulate algebraic numbers. In Section 3.6.2, we describe different algebraic tools allowing to certify the computation. Finally, in Section 3.6.3, we present an alternative to the first algorithm. As the condition of genericity is costly, we propose an algorithm partially based on subdivision which deals with algebraic curves even not in generic position.

Before explaining how the algorithm works, we will give some definitions:

Definition 8 (Critical point). *A point (α, β) of $\mathcal{C} = \mathcal{V}(f)$ is x-critical if $f(\alpha, \beta) = \frac{\partial f}{\partial y}(\alpha, \beta) = 0$.*

Definition 9 (Singular point). *A point (α, β) of $\mathcal{C} = \mathcal{V}(f)$ is singular if $f(\alpha, \beta) = \frac{\partial f}{\partial y}(\alpha, \beta) = \frac{\partial f}{\partial x}(\alpha, \beta) = 0$.*

Definition 10 (Regular point). *A point (α, β) of $\mathcal{C} = \mathcal{V}(f)$ is regular if it is not singular.*

Definition 11 (Generic position). *The curve $\mathcal{C} = \mathcal{V}(f)$ is said to be in generic position if:*

- *The leading coefficient of f with respect to y (polynomial in x) has no real roots.*
- *For every α in $\mathbb{R}$, the number of critical points with x-coordinate α is at most* 1.

So in generic position, the curve has no vertical asymptote and its x-critical points have different x-coordinates.

3.6.1 The Algorithm from a Geometric Point of View

In this section, we are going to present the geometric idea permitting to recover the topology of the curve from the computation of some particular points.

Algorithm 9 Topology of an implicit planar curve

INPUT: a polynomial $f(x,y) \in \mathbb{Q}(x,y)$ defining a curve $\mathcal{C} \subset \mathbb{R}^2$ (up to a gcd-computation and a change of variables, we can assume f is square-free and is monic in y).

1. Compute the subresultant sequence of $f(x,y)$ and $\frac{\partial f}{\partial y}(x,y)$ viewed as polynomials in y.
2. Compute the x-critical points $\{P_i = (\alpha_i, \beta_i)\}$.
3. Check that the curve is in generic position (see Section 3.6.2) and if it is not we perform a random change of variables and restart from step 1.
4. For each critical point $P_i = (\alpha_i, \beta_i)$, compute the number of regular points with x-coordinate α_i which are above and below P_i using Sturm sequences.
5. Compute the number of arcs above a value between two successive abscissas α_i, α_{i+1}, which is constant. It can be done for example choosing a rational x-coordinate a between α_i and α_{i+1} and computing the number of real solutions of $f(a,y) = 0$ using Sturm sequences. Then we compute numerical approximations of those different points.
6. Construct the segments connecting the points computed before. For that purpose, consider a section $x = \alpha_i$, i.e. all the points of the curve with abscissas α_i and the next section $x = \alpha_{i+1}$ (see Figure 3.1).
   - Choose a rational point $a \in \,]\alpha_i, \alpha_{i+1}[$ and compute the section corresponding to $x = a$.
   - In the section $x = \alpha_i$, compute the number λ_i of points of $\mathcal{C}$ above (α_i, β_i) and μ_i the number of points below.
   - Connect the λ_i points above (α_i, β_i) with the λ_i points of largest y-coordinate of the section $x = a$, respecting the order on the y-coordinate.
   - Connect the μ_i points under (α_i, β_i) with the μ_i points of smaller y-coordinate of the section $x = a$, respecting the order on the y-coordinate. After that, connect the remaining points of the section $x = a$ to the critical point (α_i, β_i).

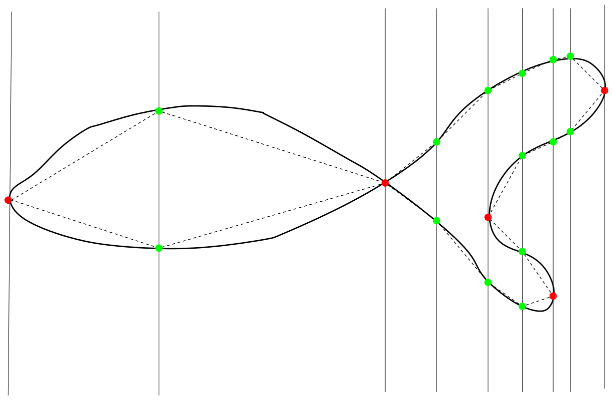

Fig. 3.1. Connections

3.6.2 Algebraic Ingredients

Computing the topology will lead us to the treatment of algebraic numbers. In this section, we come back to some delicate points of the algorithm. We show how to use the algebraic tools presented in the previous section to certify these steps.

Computing the critical points:

Proposition 6. *Let f be a square-free polynomial in $\mathbb{R}[x,y]$ of degree d in y, such that $\mathcal{C} = \mathcal{V}(f)$ is in generic position.*
Let Sr_j and sr_j denote the j^{th} subresultant and subresultant coefficient of f and $\frac{\partial f}{\partial y}$ (considered as polynomials in y). If (α, β) is a critical point of $\mathcal{C}$, then there exists $k \in \mathbb{N}^*$ such that:

$$\mathrm{sr}_0(\alpha) = 0, \ldots, \mathrm{sr}_{k-1}(\alpha) = 0, \mathrm{sr}_k(\alpha) \neq 0.$$

Moreover, we have: $\beta = \frac{-1}{k} \frac{\mathrm{sr}_{k,k-1}(\alpha)}{\mathrm{sr}_k(\alpha)}$, where $\mathrm{sr}_{k,k-1}(x)$ denotes the coefficient of y^{k-1} in $\mathrm{Sr}_{k-1}(x,y)$.

So, we observe that in generic position, the y-coordinate β of a critical point (α, β) is a rational expression of α.
We define inductively a family of polynomials $\Gamma_k(x)$:

$$\begin{array}{ll}
\Phi_0(x) = \frac{\mathrm{sr}_0(x)}{\gcd(\mathrm{sr}_0(x), \mathrm{sr}_0'(x))} & \\
\Phi_1(x) = \gcd(\Phi_0(x), \mathrm{sr}_1(x)) & \Gamma_1 = \frac{\Phi_0(x)}{\Phi_1(x)} \\
\Phi_2(x) = \gcd(\Phi_1(x), \mathrm{sr}_2(x)) & \Gamma_2 = \frac{\Phi_1(x)}{\Phi_2(x)} \\
\Phi_3(x) = \gcd(\Phi_2(x), \mathrm{sr}_3(x)) & \Gamma_3 = \frac{\Phi_2(x)}{\Phi_3(x)} \\
\vdots & \vdots \\
\Phi_0(x) = \gcd(\Phi_{d-2}(x), \mathrm{sr}_{d-1}(x)) & \Gamma_{d-1} = \frac{\Phi_{d-2}(x)}{\Phi_{d-1}(x)}
\end{array}$$

We deduce that the square-free part $\Phi_0(x)$ of $\mathrm{sr}_0(x)$ can be written as $\Phi_0 = \Gamma_1(x).\Gamma_2(x)\cdots\Gamma_{d-1}(x)$. If (α, β) is a critical point of $\mathcal{C}$ in generic position, then α is a root of Φ_0. It is in fact the root of one and only one of the Γ_i. And if α is a root of Γ_k, we have that $\mathrm{sr}_1(\alpha) = 0, \ldots, \mathrm{sr}_{k-1}(\alpha) = 0, \mathrm{sr}_k(\alpha) \neq 0$. Thus $\gcd(f(\alpha, y), \frac{\partial f}{\partial y}(\alpha, y)) = \mathrm{Sr}_k(\alpha, y)$ and β is the only root of $\mathrm{Sr}_k(\alpha, y)$ (with multiplicity k).

Generic position: An important operation that we have to perform, is to check that $\mathcal{C}$ is in generic position. If α_i is the x-coordinate of a critical point, α_i is the root of a certain Γ_k and we note this root α_i^k. Then:

Proposition 7. *The curve $\mathcal{C}$ is in generic position if and only if for every α_i^k, the polynomial $\mathrm{Sr}_k(\alpha_i^k, y)$ has only one distinct root which is $\beta_i^k = \frac{-1}{k}\frac{\mathrm{sr}_{k,k-1}(\alpha)}{\mathrm{sr}_k(\alpha)}$.*

We have to check that $\mathrm{Sr}_k(\alpha_i^k, y)$ has only one distinct root where α_i^k is defined by a pair $(\Gamma_k,]a, b[)$, so that α_i^k is the only root of Γ_k in $]a, b[$. We compute a Sturm sequence for $\mathrm{Sr}_k(\alpha_i^k, y)$. It is a family of polynomials in $\mathbb{Q}[\alpha_i][y]$. So to apply Sturm's theorem, we need to compute signs of polynomials expressions in α_i^k. Such signs can be computed, using another Sturm sequence.

Number of points above and under a critical one: Assume $\mathcal{C}$ is in generic position. If $P_\alpha = (\alpha, \beta)$ is a critical point of $\mathcal{C}$, then we need to compute the number of regular points with x-coordinate α which are above and under P_α. We can assume that α is a root of Γ_k. Then y-coordinates of the regular points with abscissas α are the roots of the polynomial $F_k(\alpha, \beta, y) = \frac{f(\alpha,y)}{(y-\beta)^k}$. The coefficients of $F_k(\alpha, \beta, y)$ can be computed in an inductive way [182]. So as β is a rational expression of α we obtain the coefficients of $F_k(\alpha, \beta, y)$ as rational expressions in α. We determine the number of roots of this polynomial such that $y - \beta > 0$ (resp. $y - \beta < 0$). This can be computed again using Sturm sequences [103].

3.6.3 How to Avoid Genericity Conditions

We have seen that to check the genericity position of a curve, we had to compute several Sturm sequences, which can be costly. An important improvement would be able to deal with curves even not in generic position.

1. We compute the two resultants $\mathrm{Res}_y(f, \partial_y f)$ and $\mathrm{Res}_x(f, \partial_y f)$. This allows us to compute isolating boxes, containing at most one x-critical point of the curve. Let $B =]a, b[\times]c, d[$ be a box which contains an x-critical point (α, β). First, we refine (or delete) the box so that there is one and only one point with abscissa α in B. For that, we compute the Sturm sequence of $f(\alpha, y)$ and compute the number of changes of sign on $]c, d[$. We refine the box until the number of changes is at most one (if it is 0, the box is empty and we delete it).

2. Assume $P_{\alpha,1},\ldots,P_{\alpha,k}$ are the x-critical points with abscissas α sorted according to their y-coordinate. After the first step $P_{\alpha,i}$ is isolated in a box $]a_i,b_i[\times]c_i,d_i[$. We compute the Sturm sequence of $f(\alpha,y)$ and $\partial_y f(\alpha,y)$ and compute the number n_k of points with y-coordinate bigger than d_k. Then n_k is the number of points above $P_{\alpha,k}$, $n_{k-1}-n_k-1$ is the number of points between $P_{\alpha,k-1}$ and $P_{\alpha,k}$, …,n_1-n_2-1 is the number of points between $P_{\alpha,1}$ and $P_{\alpha,2}$. At last we compute the number of points with y-coordinate smaller than c_1. We have a family of boxes corresponding to x-critical points with abscissas α. Up to refinement, we assume all the boxes have the same x-coordinates a, b (the boxes are of the type $]a,b[\times]c,d[$). What we want, is to refine the boxes so that if the box is $B=]a,b[\times]c,d[$, then the curve $\mathcal{C}$ does not intersect the sides $[(a,c),(b,c)]$ and $[(a,d),(b,d)]$. For that, we compute the Sturm sequences of $f(x,c)$ and $f(x,d)$ and we refine the size of the box until the number of sign changes (for the interval $]a,b[$) is 0.
3. The next step consists of computing the number of points of intersection of $\mathcal{C}$ with the side $[(a,c),(a,d)]$ (resp. $[(b,c),(b,d)]$). This is done by computing the Sturm sequences of $f(a,y)$ and $\partial_y f(a,y)$ (resp. $f(b,y)$ and $\partial_y f(b,y)$) on $]c,d[$. We compute the number of points in an intermediate section $x=a\in\mathbb{Q}$ (using the Sturm Sequence of $f(x,y),\partial_y f(x,y)$ at $x=a$.
4. The connections of the different points is made similarly as in the generic case.

3.7 Topology of 3d Implicit Curves

We consider here an implicit curve in an affine space of dimension 3. By definition, it is an algebraic variety $\mathcal{C}_{\mathbb{C}}=\mathcal{V}(f_1,\ldots,f_m)$ ($f_i\in\mathbb{R}[x,y,z]$) of dimension 1 in $\mathbb{C}^3$. We denote by $I(\mathcal{C}_{\mathbb{C}})\subset\mathbb{R}[x,y,z]$, the ideal of the curve $\mathcal{C}_{\mathbb{C}}$ (that is the set of polynomials which vanish on $\mathcal{C}_{\mathbb{C}}$) and by $g_1,\ldots,g_s\in\mathbb{R}[x,y,z]$ a set of generators: $I(\mathcal{C}_{\mathbb{C}})=(g_1,\ldots,g_s)$. By Hilbert's Nullstellensatz [104, 199], we have $I(\mathcal{V}(f_1,\ldots,f_k))=\sqrt{I}\subset\mathbb{R}[x,y,z]$. It can be proved [137, 199], that 3 polynomials g_1, g_2, $g_3\in\mathbb{R}[x,y,z]$ are enough to generate $I(\mathcal{C}_{\mathbb{C}})$.

For simplicity, we will consider here that the curve is described as the intersection of two surfaces $P_1(x,y,z)=0$, $P_2(x,y,z)=0$, with $P_1,P_2\in\mathbb{R}[x,y,z]$. We assume that the gcd of P_1 and P_2 in $\mathbb{R}[x,y,z]$ is 1, so that $\mathcal{V}(P_1,P_2)=\mathcal{C}_{\mathbb{C}}$ is of dimension 1, and all its irreducible components are of dimension 1. We are interested in describing the topology of the real part

$$\mathcal{C}_{\mathbb{R}}=\{(x,y,z)\in\mathbb{R}^3,\ P_1(x,y,z)=0,\ P_2(x,y,z)=0\},$$

that we will denote hereafter by $\mathcal{C}$.

In this section, we assume moreover that $I(\mathcal{C})=(P_1,P_2)$ or equivalently that (P_1,P_2) *is a reduced ideal*, that is equal to its radical: $(P_1,P_2)=\sqrt{(P_1,P_2)}$.

We will not consider examples such as $P_1 = x^2+y^2-1, P_2 = x^2+y^2+z^2-1$, where $(P_1, P_2) = (x^2+y^2-1, z^2)$ and $I(\mathcal{C}) = (x^2+y^2-1, z)$, so that the curve $\mathcal{C}$ is defined "twice" by the equations $P_1 = 0, P_2 = 0$ (the two surfaces intersect tangentially along $\mathcal{C}$). Such a property can be tested by projecting into a generic direction and testing if the equation computed from the resultant of P_1, P_2 is square-free, or by more general methods such as computing the radical of (P_1, P_2) [189].

The general idea behind the algorithm that we are going to describe is as follows: we use a sweeping plane in a given direction (say parallel to the (y, z) plane) to detect the critical positions where *something* happens. We also compute the positions where something happens in projection on the (x, y) and (x, z) plane. Then, we connect the points of the curve of $\mathcal{C}$ on these critical planes. This yields a graph of points, connected by segments, with the same topology as the curve $\mathcal{C}$.

3.7.1 Critical Points and Generic Position

In this section, we make more precise what we mean by the points where *something* happens. These points will be called hereafter critical points.

Definition 12. *Let $I(\mathcal{C}) = (g_1, g_2, \ldots, g_s)$ and let M be the $s \times 3$ Jacobian matrix with rows $\partial_x g_i, \partial_y g_i, \partial_z g_i$.*

- *A point $p \in \mathcal{C}$ is* regular *(or smooth) if the rank of M evaluated at p is* 2.
- *A point $p \in \mathcal{C}$ which is not regular is called* singular.
- *A point $p = (\alpha, \beta, \gamma) \in \mathcal{C}$ is x*-critical *(or critical for the projection on the x-axis) if the curve $\mathcal{C}$ is tangent at this point to a plane parallel to the (y, z)-plane i.e., the multiplicity of intersection of the plane with $I(\mathcal{C})$ at p is greater or equal to 2. The corresponding α is called an x*-critical value.

A similar definition applies to the orthogonal projection onto the y and z axis or onto any line of the space. Notice that a singular point is critical for any direction of projection.

If $I(\mathcal{C}) = (P_1, P_2)$, then the x-critical points are the solutions of the system

$$P_1(x, y, z) = 0,\ P_2(x, y, z) = 0,\ (\partial_y P_1 \partial_z P_2 - \partial_y P_2 \partial_z P_1)(x, y, z) = 0. \quad (3.1)$$

In the case of a planar curve defined by $P(x, y) = z = 0$, with $P(x, y)$ square-free so that $I(\mathcal{C}) = (P(x, y), z)$, this yields the following definitions: a point (α, β)

- is *singular* if $P(\alpha, \beta) = \partial_x P(\alpha, \beta) = \partial_y P(\alpha, \beta) = 0$.
- is *x-critical* if $P(\alpha, \beta) = \partial_y P(\alpha, \beta) = 0$.

This allows us to describe the genericity condition that we require for the curve $\mathcal{C}$, in order to be able to apply the algorithm:

Definition 13 (Generic position). *Let*

$$N_x(\alpha) = \#\{\, (\beta,\gamma) \in \mathbb{R}^2 \mid (\alpha,\beta,\gamma) \textit{ is an } x\textit{-critical point of } \mathcal{C} \,\}.$$

We say that $\mathcal{C}$ is in generic position for the x-direction, if

- $\forall \alpha \in \mathbb{R}$, $N_x(\alpha) \leqslant 1$, *and*
- *there is no asymptotic direction of $\mathcal{C}$ parallel to the (y,z)-plane.*

By a random change of variables, the curve can be put in a generic position. In practice, instead of changing the variables, we may choose a random direction for the sweeping plane.

3.7.2 The Projected Curves

The algorithm that we are going to describe, uses the singular points of the projection of $\mathcal{C}$ onto the (x,y) and (x,z)-planes. We denote by $\mathcal{C}'$ (resp. $\mathcal{C}''$) the projection of the curve $\mathcal{C}$ onto the (x,y) (resp. (x,z))-plane. The equation of the curve $\mathcal{C}'$ is obtained as follows. We decompose the polynomials P_1, P_2 in terms of the variable z:

$$\begin{aligned} P_1(x,y,z) &= a_{d_1}(x,y)z^{d_1} + ... + a_0(x,y) \\ P_2(x,y,z) &= b_{d_2}(x,y)z^{d_2} + ... + b_0(x,y) \end{aligned}$$

with $\mathrm{a}_{d_1}(x,y) \neq 0$ and $\mathrm{b}_{d_2}(x,y) \neq 0$. Then, the resultant polynomial

$$G(x,y) = \mathrm{Res}_z(P_1, P_2)$$

vanishes on the projection of the curve $\mathcal{C}$ on the plane (x,y). Conversely, by the resultant theorem [233, 138] (see also Section 3.4.1), $a_{d_1}(x,y)$ and if the gcd $c(x,y)$ of $a_{d_1}(x,y)$ and $b_{d_2}(x,y)$ in $\mathbb{R}[x,y]$ is 1 then $a_{d_1}(x,y)$ and $b_{d_2}(x,y)$ do not vanish simultaneously on a component of dimension 1 of the projection $\mathcal{C}'$ of the curve $\mathcal{C}$. So $G(x,y) = 0$ defines $\mathcal{C}'$ and a finite number of additional points. If it's not the case, G is a non-trivial multiple of the implicit equation of $\mathcal{C}'$. Such a situation can be avoided, by a linear change of variables. Nevertheless, since the critical points of the curve defined by $G(x,y) = 0$ contains the critical points of $\mathcal{C}'$, we will see hereafter that this change of variables is not necessary.

Notice, that $G(x,y)$ is not necessarily a square-free polynomial. Consider for instance the case $P_1 = x^2 + y^2 - 1, P_2 = x^2 + y^2 + z^2 - 2$, where $G(x,y) = (x^2 + y^2 - 1)^2$. In this case, there are generically two (complex) points of $\mathcal{C}$ above a point of $\mathcal{C}'$.

We can easily compute the gcd of $G(x,y)$ and $\partial_y G(x,y)$ in order to get the square-free part $g(x,y) = G(x,y)/\gcd(G(x,y), \partial_y G(x,y))$ of $G(x,y)$.

Similarly, for the projection $\mathcal{C}''$ of $\mathcal{C}$ on the (x,z)-plane, we compute

$$H(x,z) = \mathrm{Res}_y(P_1, P_2),$$

and its square-free part $h(x,z)$ from the gcd of $H(x,z)$ and $\partial_z H(x,z)$. The equation $h(x,z)=0$ defines a curve which is exactly $\mathcal{C}''$, if the gcd of the leading components of P_1, P_2 in y is 1. Its set of singular points contains those of $\mathcal{C}''$.

In order to analyze locally the projection of the curve $\mathcal{C}$, we recall the following definition:

Definition 14. *[333] Let X be an algebraic subset of $\mathbb{R}^n$ and let p be a point of X. The tangent cone at p to X is the set of points u in $\mathbb{R}^n$ such that there exists a sequence of points x_k of X converging to p and a sequence of real numbers t_k such that $\lim_{k\to+\infty} t_k(x_k-p)=u$.*

Notice, that at a smooth point of $\mathcal{C}$, the tangent cone is a line.

Proposition 8. *Let $p'=(\alpha,\beta)$ be an x-critical point of $\mathcal{C}'$, which is not singular. Then α is the x-coordinate of an x-critical point of $\mathcal{C}$.*

3.7.3 Lifting a Point of the Projected Curve

The problem we want to tackle here is the following: Assume we are given two surfaces defined by two implicit equations $P_1=0$ and $P_2=0$. Let us consider the projection of the curve of intersection of the two surfaces on the (x,y)-plane.

> *Starting from a point (x_0,y_0) of the projected curve, how can we find (a numerical approximation of) the z-coordinate of the point(s) above (x_0,y_0)?*

We note $p(z)=P_1(x_0,y_0,z)$, $q(z)=P_2(x_0,y_0,z)$ and $d=\deg(p)$, $d'=\deg(q)$.

Consider the Sylvester submatrix $\mathrm{Syl}_1(x_0,y_0)$ of the mapping

$$\begin{array}{ccc} \mathbb{R}[z]_{d'-2}\oplus\mathbb{R}[z]_{d-2} & \longrightarrow & \mathbb{R}[z]_{d+d'-2} \\ (u,v) & \mapsto & p\,u+q\,v \end{array}$$

If ξ is a common root of p and q then $(1,\xi,\ldots,\xi^{d+d'-2})$ is in the kernel of the transpose of $\mathrm{Syl}_1(x_0,y_0)$. If we assume that $\mathrm{Syl}_1(x_0,y_0)$ is of maximal rank, and if Δ_i denotes the minor of $\mathrm{Syl}_1(x_0,y_0)$ obtained by removing the row i, then the (non-zero) vector $[\Delta_1,-\Delta_2,\ldots,(-1)^{d+d'-1}\Delta_{d+d'-1}]$ is in the kernel of the transpose of $\mathrm{Syl}_1(x_0,y_0)$. Thus $(1,\xi,\ldots,\xi^{d+d'-2})$ and $[\Delta_1,-\Delta_2,\ldots,(-1)^{d+d'-1}\Delta_{d+d'-2}]$ are linearly dependent. We deduce that $\xi=-\frac{\Delta d+d'-1}{\Delta d+d'-2}=-\frac{S_{1,0}(x_0,y_0)}{S_{1,1}(x_0,y_0)}$.

This method allows us to lift a point on $\mathcal{C}$, if there is only one point above (x_0,y_0), but it can be generalized when there are several points above. This generalization is closely related to the subresultant construction of univariate polynomials [326]. Here we want to exploit linear algebra tools from a numerical perspective. The aim is to make the matrix of multiplication by z in the quotient algebra $\mathbb{R}[z]/(P_1(x_0,y_0,z),P_2(x_0,y_0,z))$ appear, in order to compute its eigenvalues which yields z-coordinate of the points above (x_0,y_0) [138].

We proceed as follows: Given a point (x_0, y_0) of the projected curve $\mathcal{C}'$, we construct the Sylvester matrix associated to $P(z)$, $q(z)$. By construction, the columns of this matrix are $P, z\,P, \ldots, z^{d'-1}\,P, q, z\,q, \ldots, z^{p-1}\,q$, written in the basis $1, z, \ldots, z^{d+d'-1}$. Assume that the kernel of the transposed Sylvester matrix $\mathrm{Syl}(x_0, y_0)$ has dimension d and is generated by $\Lambda_1, \ldots, \Lambda_d$.
By transposition, we can interpret the Λ_i $(i = 1 \ldots d)$ as linear forms over $\mathbb{Q}_{d+d'-1}[z]$ vanishing on $P, z\,P, \ldots, z^{d'-1}\,p, q, z\,q, \ldots, z^{d-1}\,q$. We can extend the Λ_i over $\mathbb{R}[z]$, considering that these forms vanish over all the ideal generated by p and q. So they can be considered as elements of the dual of $\mathcal{A} = \mathbb{R}[z]/(p(z), q(z))$. As the linear forms Λ_i are independent, they also form a basis of this dual space. The coefficients of Λ_i in the dual basis $(1^*, \ldots, (z^{d-1})^*)$ of the monomial basis $\{1, z, \ldots, z^{d-1}\}$ of $\mathcal{A}$ are $[\Lambda_i(1), \Lambda_i(z), \ldots, \Lambda_i(z^{d-1})]$. By definition of the transposed operator, for any $a \in \mathcal{A}$, $M^t(\Lambda_i)(a) = \Lambda_i(M_z(a)) = \Lambda_i(z\,a)$. Thus we have the relation:

$$\begin{pmatrix} \Lambda_1(z) & \ldots & \Lambda_d(z) \\ \vdots & & \vdots \\ \Lambda_1(z^d) & \ldots & \Lambda_d(z^d) \end{pmatrix} = M_z^t \begin{pmatrix} \Lambda_1(1) & \ldots & \Lambda_d(1) \\ \vdots & & \vdots \\ \Lambda_1(z^{d-1}) & \ldots & \Lambda_d(z^{d-1}) \end{pmatrix}$$

where M_z is the operator of multiplication by z in $\mathbb{R}[z]/(p(z), q(z))$. As $d = \dim\ker(\mathrm{Syl}(x_0, y_0)) = \dim \mathcal{A}$, and as $(1, z, ..., z^{d-1})$ form a basis of the quotient space, the matrix

$$\begin{pmatrix} \Lambda_1(1) & \ldots & \Lambda_d(1) \\ \vdots & & \vdots \\ \Lambda_1(z^{d-1}) & \ldots & \Lambda_d(z^{d-1}) \end{pmatrix}$$

is invertible. We deduce that computing the generalized eigenvalues of the previous matrices yields the eigenvalues of the operator M_z of multiplication by z in $\mathcal{A}$, that is the z-coordinate of the points above (x_0, y_0).

We summarize the algorithm here:

Algorithm 10 Lifting the projection

- Compute the Sylvester matrix $S = \mathrm{Syl}(x_0, y_0)$.
- Compute a basis $\Lambda_1, \ldots, \Lambda_d$ of the kernel of S^t.
- Extract the submatrix A_0 of the coordinates of $\Lambda_1, \ldots, \Lambda_d$ corresponding to the evaluations in $1, \ldots, z^{d-1}$.
- Extract the submatrix A_1 of the coordinates of $\Lambda_1, \ldots, \Lambda_d$ corresponding to the evaluations in $z, \ldots, z^d$.
- Compute the generalized eigenvalues of A_1 and A_0 and output the corresponding z-coordinates of the point above (x_0, y_0).

The last step can be replaced by the computation of $\det(A_1 - z\,A_0)$ and a univariate root finding step.

3.7.4 Computing Points of the Curve above Critical Values

In this section, we are going to describe how we check the genericity condition and how we compute a finite set of points, which will allow us to deduce the topology of $\mathcal{C}$.

First, we check that there is no asymptotic direction parallel to the (y, z)-plane, by testing if the curve $\mathcal{C}$ has a point at infinity in the plane $x = 0$. This is done by checking if the system

$$\frac{P_1^\top}{\Delta}(0, y, z) = \frac{P_2^\top}{\Delta}(0, y, z) = 0$$

has a non-trivial solution, where $P^\top$ is the homogeneous component of highest degree of a polynomial P and $\Delta = \gcd(P_1^\top, P_2^\top)$. It reduces to computing the projective resultant of these two homogeneous polynomials. Since the number of asymptotic directions of $\mathcal{C}$ is finite, by a generic linear change of variables, we can avoid the cases where $\mathcal{C}$ has an asymptotic direction parallel to the (y, z)-plane.

Next, we compute the x-critical points of $\mathcal{C}$ by solving the system (3.1), using Algorithm 8. This computation allows us to check that the system is zero-dimensional and that the x-coordinates of the real solutions are distinct. If this is not the case, we perform a generic change of coordinates.

The cases for which we have to do a change of coordinates are those where a component of $\mathcal{C}$ is in a plane parallel to (y, z) or where a plane parallel to (y, z) is tangent to $\mathcal{C}$ in two distinct points. Such cases are avoided by a generic change of coordinates.

We denote by $\Sigma_0 = \{\sigma_1^0, \ldots, \sigma_{k_0}^0\}$ the x-coordinates of the x-critical points: $\sigma_1^0 < \cdots < \sigma_{k_0}^0$.

Next, we compute the singular points of $\mathcal{C}'$ as (a subset of) the real solutions of the system

$$g(x, y) = 0, \partial_x g(x, y) = 0, \partial_y g(x, y) = 0, \tag{3.2}$$

and of $\mathcal{C}''$, as (a subset of) the real solutions of

$$h(x, z) = 0, \partial_x h(x, z) = 0, \partial_z h(x, z) = 0. \tag{3.3}$$

We denote by $\Sigma_1 = \{\sigma_1^1, \ldots, \sigma_{k_1}^1\}$ the x-coordinates of these singular points: $\sigma_1^1 < \cdots < \sigma_{k_1}^1$.

An important property of the projected curves $\mathcal{C}'$ and $\mathcal{C}''$, that will be used in the algorithm, is the following:

Proposition 9. *The arcs of the curve $\mathcal{C}'$ (resp. $\mathcal{C}''$) above $]\sigma_i, \sigma_{i+1}[$ do not intersect.*

Let $k(x)$ be the square-free part of $\mathrm{Res}_y(g(x, y), \partial_y g(x, y)) * \mathrm{Res}_z(h(x, z), \partial_z h(x, z))$. We consider the zero-dimensional system:

$$\begin{cases} k(x) &= 0 \\ P(x,y,z) &= 0 \\ Q(x,y,z) &= 0 \end{cases}$$

Computing generalized normal forms and computing a Rational Univariate Resolution of this system (see Section 3.5), we obtain a representation of the solutions of this system as:

$$\begin{cases} f_0(T) = 0 \\ x \quad = f_1(T) \\ y \quad = f_2(T) \\ z \quad = f_3(T) \end{cases}$$

3.7.5 Connecting the Branches

The approach that we are going to describe now for the branch connection, can be seen as an extension of the approach of [188, 182] to the three-dimensional case.

The previous step yields a sequence of strictly increasing values

$$\Sigma = \{\sigma_1, \ldots, \sigma_l\},$$

such that above $]\sigma_i, \sigma_{i+1}[$, the branches of $\mathcal{C}$ are smooth and the arcs of $\mathcal{C}'$, $\mathcal{C}''$ do not intersect. We will use this property to connect the points of $\mathcal{C}$ above the values σ_i. Notice that Proposition 9 is still true if we refine the sequence $\sigma_1, \ldots, \sigma_l$. In particular, it is valid if we consider the x-coordinates of the singular points of a curve, defined by a multiple of the equation of $\mathcal{C}'$ (resp. $\mathcal{C}''$). It is also valid, if we insert new values in between these critical values: $\delta_0 < \sigma_1 < \mu_1 < \cdots < \sigma_l < \delta_1$, where $\mu_i := \frac{\sigma_i + \sigma_{i+1}}{2}$ for $i = 0, \ldots, l-1$, and δ_0, δ_1 are any value such that $]\delta_0, \delta_1[$ contains Σ. We denote by

$$\alpha_0 < \cdots < \alpha_m$$

this new refined sequence of values and by L_i, the set of points on $\mathcal{C}$ above α_i, for $i = 0, \ldots, m$. These points are computed, either

- by substituting $x = \alpha_i$ and solving the 2-dimensional system $P_1(\alpha_i, y, z) = 0$, $P_2(\alpha_i, y, z) = 0$.
- or by computing the points of $\mathcal{C}'$ above α_i and by lifting them to $\mathcal{C}$ (Algorithm 10).

This construction implies the following lemma, which is used in the next theorem, in order to describe how the computed points have to be connected:

Lemma 3. *Two distinct points of a regular section of $\mathcal{C}$ with the same y-coordinate (resp. z-coordinate) are connected to two points of the next section, with the same y-coordinate (resp. z-coordinate) or to a critical point.*

Theorem 8. *Under the genericity condition of Definition 13, the curve $\mathcal{C}$ connects the points L_i to the points L_{i+1}, only in one way.*

To summarize, the connection of the branches from one plane section of $\mathcal{C}$ to the next one, is performed as follows:

Algorithm 11 Connecting the branches

If there is no x-critical point in L_i and possibly an x-critical point c of $\mathcal{C}$ in L_{i+1}, do the following:

1. Decompose L_{i+1} into the subsets $V'_1, \ldots, V'_k$ of the points with the same y-coordinate, listed by increasing y. Let $s_j = |V'_j|$.
2. Compute the index j_0 such that $c \in V'_{j_0}$. Decompose L_i into the subsets $V_1, \ldots, V_k$ in the following way:
   - For $j > j_0$, V_j is the set of s_j greatest points for the lexicographic order with $x > y > z$, among $L_i - \cup_{l>j} V_l$.
   - For $j < j_0$, V_j is the set of s_j smallest points for the lexicographic order, among $L_i - \cup_{l<j} V_l$.
   - V_{j_0} is the remaining set of points $L_i - \cup_{l \neq j_0} V_l$.
3. For $j \neq j_0$ connect the points of V_j to the points of V'_j, according to there z-coordinates, by segments.
4. For $j = j_0$, let A'_{j_0} (resp. B'_{j_0}) be the set of regular points of V'_{j_0}, with z-coordinate $< z(c)$ (resp. $> z(c)$).
   - Connect the $|A'_{j_0}|$ points of smallest z-coordinates in V_{j_0} to the points in A'_{j_0}, according to their z-coordinate, by segments.
   - Connect the $|B'_{j_0}|$ points of greatest z-coordinates in V_{j_0} to the points in B'_{j_0}, according to their z-coordinate, by segments.
   - Connect the remaining points in V_{j_0} to c, by segments.

If there is a x-critical point of $\mathcal{C}$ in L_i, exchange the role of L_i and L_{i+1} in the previous steps.

Proposition 10. *Assume that we are in a generic position. Then the topology of the curve above the segment $[\alpha_i, \alpha_{i+1}]$ is the same as the set of segments produced by the Algorithm 11.*

3.7.6 The Algorithm

We summarize the complete algorithm below:

Remark 1. This algorithm can be easily adapted to the computation of the topology of $\mathcal{C}$ in a box (resp. bounded domain), by considering the points on the border of the box (resp. domain) as x-critical points.

Remark 2. By a generic change of variables, the set of x-coordinates of the x-critical points of $\mathcal{C}'$ will contain those of $\mathcal{C}$ and the resolution of the system

Algorithm 12 Topology of the curve $\mathcal{C} \subset \mathbb{R}^3$ defined by two polynomials

INPUT: polynomials $P_1(x, y, z), P_2(x, y, z) \in \mathbb{Q}[x, y, z]$.

- Compute the x-critical points of $\mathcal{C}$ and their x-coordinates $\Sigma := \{\sigma_1^0, \ldots, \sigma_k^0\}$ with $\sigma_1^0 < \cdots < \sigma_k^0$.
- Check the generic position; If the curve is not in a generic position, apply a random change of variables and restart from the first step.
- Compute the square-free part $g(x, y)$ of $\mathrm{Res}_z(P_1, P_2)$.
- Compute the square-free part $h(x, z)$ of $\mathrm{Res}_y(P_1, P_2)$.
- Compute the singular points of the curves $g(x, y) = 0$ and $h(x, z) = 0$ and insert their x-coordinate in Σ.
- Compute the $\mu_i, \delta_0, \delta_1$ and the ordered sequence $\alpha_1 < \cdots < \alpha_l$. Above each α_i for $i = 1, \ldots, l$, compute the set of points L_i on the curve $\mathcal{C}$.
- For each $i = 0, \ldots, l-1$, connect the points L_i to those of L_{i+1} by Algorithm 11.

OUTPUT: the graph of 3D points connected by segments, with the same topology as the curve $\mathcal{C}$.

(3.1) can be replaced by the computation of the x-critical points of $\mathcal{C}'$ and by a lifting operation on $\mathcal{C}$. This allows us to treat unreduced curves, such that $I(\mathcal{C}) \neq (P_1, P_2)$, by using only the square-free part of $G(x, y)$ and $H(x, z)$. However the verification, a posteriori, of the correctness of the result is more delicate.

3.8 Software

The algebraic-geometric computation that we have seen, is based on an exact arithmetic model, which may involve large size numbers. The library GMP [6] provides tools for manipulating integer and rational numbers of arbitrary size. There are also tools for manipulating floating point numbers of arbitrary size. If control over the precision and the rounding mode of each operation are needed, one can also use MPFR, developed in the same spirit [10].

CGAL [2] provides interval arithmetic, as well as a simple multi-precision floating point type able to evaluate signs of polynomials over floating point input exactly, it also interfaces with external libraries such as GMP and LEDA for multi-precision computations. CGAL also provides generic mechanisms to apply filtering techniques to its own kernel of geometric primitives as well as user defined primitives [277, 278, 65]. Finally, CGAL provides some fine-tuned statically filtered predicates [247] for Delaunay triangulations and other classical geometric algorithms.

BOOST [1] provides a generic implementation of interval arithmetic which can be used in many other contexts [67]. Finally, MPFI [9] provides interval arithmetic with multiple-precision bounds based on MPFR [10]. Other resources on interval computations can be found at http://www.cs.utep.edu/interval-comp/main.html, together with many other interval arithmetic libraries.

There are several implementations of algebraic numbers which use the separation bound approach for comparison. The LEDA library [72] implements real algebraic root expressions (without the $\diamond$-operator) using the bound given above. The improved separation bound [280, 301] and the diamond operator are implemented in [170]. They will be included in the LEDA library soon.

The CORE library [3] implements real algebraic expression using several different separation bounds.

A package of the library SYNAPS [13] is devoted to the treatment of algebraic numbers represented by an isolating interval and a square-free polynomial of $\mathbb{Z}[x]$. The specialization for small degree developed by E. Tsigaridas and I. Emiris is available there. The isolation and comparison of small degree algebraic numbers are given by precomputed formulas. For higher degrees, isolation and comparison methods based on the Bernstein subdivision method or on the Sturm-Habicht computations are also implemented.

Regarding multivariate polynomial systems solving, one can also find in SYNAPS [13] solvers based on normal form computation (developed by Ph. Trébuchet) or subdivision solvers (developed by J.P. Pavone), which are very interesting for filtering purposes. Implementation of polynomial solvers based on Gröbner computation are also available in GBRS [5], or SINGULAR [12].

Regarding the topology of implicit curves, implementations of the sweeping algorithm and of other methods based on subdivision techniques are available in SYNAPS [13].