Journal of Mathematical Imaging and Vision, ??, 1–32 (??)

Algebraic Geometry and Computer Vision: Polynomial Systems, Real and Complex Roots

S. PETITJEAN

petitjea@loria.fr

CRIN - CNRS & INRIA *Lorraine, Bâtiment* LORIA*, BP 239, 54506 Vandœuvre-les-Nancy cedex, France*

Received 2/25/97; Revised 6/4/97

Editors: Prof. G.X. Ritter

Abstract. We review the different techniques known for doing exact computations on polynomial systems. Some are based on the use of Gröbner bases and linear algebra, others on the more classical resultants and its modern counterparts. Many theoretical examples of the use of these techniques are given. Furthermore, a full set of examples of applications in the domain of artificial vision, where many constraints boil down to polynomial systems, are presented. Emphasis is also put on very recent methods for determining the number of (isolated) real and complex roots of such systems.

Keywords: Polynomial systems, symbolic computation, resultants, computer vision applications

1. Introduction

In recent years, the search for efficient algorithms for solving problems related to systems of polynomial equations has received renewed attention because of their demonstrated importance to a variety of problems of both practical and theoretical interest. The need to count the number of real solutions or to find exact or approximate solutions to such systems has arisen in a wide range of practical areas, including robotics, computational number theory and solid modelling. For instance, the "piano mover's problem" (theoretical robotics) and several parts of graphics research involve the study of semi-algebraic sets, intersection of algebraic and semi-algebraic sets, ...

"*Computer algebra is that part of computer science which designs, analyses, implements, and applies algebraic algorithms*" [16]. Computer algebra is a synonym of "symbolic computation" and "algebraic computation". It is a special form of scientific computation and it comprises a wide range of basic goals, methods and applications. In contrast to numerical computation, the emphasis is on computing with symbols representing mathematical concepts. Symbolic computation encompasses all types of exact computations, be them on differential equations, integral equations or algebraic systems.

In this paper, we review the different solutions that computer algebra and algebraic geometry have to offer for dealing with polynomial systems and examine some of their recent applications in the field of computer vision. We only concentrate here on symbolic (exact) methods, which rules out such numerical or probabilistic methods as homotopy continuation [63]. In particular, we shall be concerned with the following two application domains: polynomial system solving and *real algebraic geometry* [10, 13]. Though the boundary between the two is hard to draw, we may roughly say that real algebraic geometry is concerned with semi-algebraic sets (subsets of some real Euclidean space that are defined by polynomial equalities and inequalities) and algorithms on these sets. This means that the name does not usually cover some of the techniques of algebraic geometry working over the reals (like real intersection theory and enumerative geometry [28, 97]).

In the sequel, we focus on some of the most efficient and recent techniques for dealing with algebraic equation systems, over the reals and over the complexes. Section 2 starts by recalling which geometric reasoning problems stirred to do research on efficiently solving polynomial systems. These problems appear in first-order logical terms to be decision problems, description problems, quantifier elimination problems or mixed problems. In section 3, we look at the classical field of elimination, resultant-based and sparse resultant-based techniques. Techniques based on the use of Gröbner bases are subsequently treated in section 4 for the complex case and in section 5 those for the real case. How things may be recast as linear algebra problems is explained. In section 6, several examples of applications to problems in artificial vision are indicated within the context of algebraic computation. Some complexity issues are presented in section 7. The computational side is presented in section 8, where links are given to existing computer algebra systems and dedicated platforms. We conclude by indicating directions in which polynomial system solving should witness developments soon, like the interaction of numerical and symbolic techniques and the study of the amount of real zeros of a polynomial system.

2. Geometric reasoning problems

The original motivation for such algorithms as multivariate polynomials factorization, real root isolation, ... was their occurrence as subproblems in a much more general problem, *quantifier elimination*. We shall thus start by briefly recasting the whole thing in the context of first-order logic, following [3] (section 2.1). After that, we indicate in section 2.2 how problems in geometric reasoning fall in one of three types. The most important type of problems amounts to eliminate the quantifiers of a first-order sentence, which we describe in section 2.3. We conclude the section by giving some more examples of geometric reasoning problems in section 2.4, which provide incentives for the material reviewed in the ensuing sections.

2.1. First-order logic

Definition 1. *First-order logic*, also known as (first-order) *predicate logic*, is the language describing the truth of mathematical formulas. A *formula* or *sentence* is a sequence, which is either true or false, of symbols representing

- *terms*: mathematical expressions involving numbers, operators, functions and variables;
- *predicates*: propositional functions (*i.e.* taking any sort and number of variables as entry and having values in {true, false});
- *connectives*: operators used to combine two logical formulas;
- *quantifiers*: operators specifying for which values of a variable a formula is true. Universally quantified means "for all values" (denoted $\forall$) and existentially quantified means "there exists some value" (denoted $\exists$).

The "order" of a logic specifies what entities $\forall$ and $\exists$ may quantify over. First-order logic can only quantify over *atomic* formulas [5], *i.e.* true, false or $p(t_1, \ldots, t_n)$ with p a predicate and $t_1, \ldots, t_n$ are terms. In first-order logic, quantifiers always range over all the elements of the domain of discourse.

A set L of connectives for functions, relations and constants (*i.e.* primitive non-logical symbols) for a first-order language appropriate for the theory of the real numbers is $L = \{+, \times, =, <, 0, 1\}$, where 1 stands for true, 0 stands for false, $\times$ is the conjunction operator ($f \times g$ is true if both f and g are true), $+$ is the disjunction operator ($f + g$ is true if either or both are true), $f = g$ is true if both are true or both false and $f < g$ is true if f is false and g is true. It is often more convenient to use a larger set $L' = \{+, -, \times, <, \leq, >, \geq, \neq, 0, 1\}$, where for instance $f \neq g$ is true if either f is true and g is false or f is false and g is true, and $f \leq g$ is true if either f is false or g is true. Obviously any formula involving the symbols of L' can be rewritten in terms of the symbols of L. The terms of the language L are multivariate polynomials with rational coefficients. Let $\wedge, \vee, \neg$ be the logical symbols respectively standing for `AND`, `OR` and `NOT`.

The first-order theory of the real numbers is the collection $\mathrm{Th}(\mathbb{R})$ of all sentences of the language L which are true for the real numbers. It is known that for any sentence ϕ of L, either ϕ or $\neg\phi$ belongs to $\mathrm{Th}(\mathbb{R})$. For instance, the sentence ϕ

$$(\exists x)\ x^4 + 1 = 0$$

is such that $\neg\phi \in \mathrm{Th}(\mathbb{R})$. We use the convention that $\phi(x_1, \ldots, x_n)$ denotes a formula in which all occurrences of $x_1, \ldots, x_n$ are *free* (*i.e.*, not quantified), each x_i may or may not actually occur, and there are no free variables other than $x_1, \ldots, x_n$.

Two formulae $\phi(x_1, \ldots, x_n)$ and $\psi(x_1, \ldots, x_n)$ are said to be *equivalent* if for all real numbers $a_1, \ldots, a_n$, $\phi(a_1, \ldots, a_n)$ is true if and only if $\psi(a_1, \cdots, a_n)$ is true. In other words, they are equivalent if the sentence

$$(\forall x_1) \cdots (\forall x_n)\ [\phi(x_1, \ldots, x_n) \Leftrightarrow \psi(x_1, \ldots, x_n)]$$

belongs to $\mathrm{Th}(\mathbb{R})$.

Let E^k ($k \geq 0$) denote the k-dimensional Euclidean space (*i.e.* $\mathbb{R}^k$ with the usual topology). A subset X of E^n is *definable* if for some formula $\phi(x_1, \ldots, x_n)$ of L, X is the set of points of E^n that satisfy ϕ. We say that X is *defined* by ϕ.

2.2. Decision, description and quantifier elimination problems

Many objects used in computer vision, graphics and solid modelling are *first-order*, in the sense that they can be defined by a sentence of L. Examples are spheres, cylinders, Bézier patches and algebraic objects in general. Many problems that arise in geometric reasoning are of one of the following types:

1. *query* problem: does a collection of objects possess a certain property;

   Example 1: Let X_1 and X_2 be two first-order surfaces in E^3. Do they intersect? Let f_1 and f_2 be the equations defining X_1 and X_2. Then the formulae

   $$\phi_1(x, y, z) := [f_1(x, y, z) = 0] \text{ and } \phi_2(x, y, z) := [f_2(x, y, z) = 0]$$

   respectively define X_1 and X_2. Let

   $$\phi(x, y, z) := [\phi_1(x, y, z) \wedge \phi_2(x, y, z)].$$

   The above query can be answered by deciding if the following sentence is true or false:

   $$(\exists x)(\exists y)(\exists z)\ \phi(x, y, z). \qquad \square$$

2. *description* (*display* in the language of [3]) problem: describe an object.

   Example 2: Given first-order surfaces X_1 and X_2 of E^3 that intersect, describe their intersection. Continuing the previous example, let D be the intersection curve of X_1 and X_2. The description problem may consist of: giving the dimension of D, computing its connected components, ... $\square$

3. *constraint* problem: given a parameterised collection of objects and a property, write down the conditions on the parameters that characterise the objects having the property;

   Example 3: Let X be a first-order surface of E^3 and Y_t be a one-parameter family of first-order surfaces of E^3. Find the conditions on t such that X and Y_t intersect. If f and g_t are the equations defining X and Y_t respectively, then the problem consists in eliminating the quantifiers $\exists$ and $\forall$ from the following formula:

   $$(\exists x)(\exists y)(\exists z)\ [[f(x, y, z) = 0] \wedge [g_t(x, y, z) = 0]].$$

   $\square$

2.3. Quantifier elimination for real closed fields

Rephrased in the context of mathematical logic, the general problems of the first-order theory of the reals are well-known - at least for types (1) and (3). Type (1) is known as a *decision problem*: formally, it is the task of determining whether a particular sentence of L belongs to $\mathrm{Th}(\mathbb{R})$. Type (3) is known as a *quantifier elimination problem* (that we shall refer to as QE in what follows).

The fundamental result for the quantifier elimination problem for $\mathrm{Th}(\mathbb{R})$ was given by Tarski:

Theorem 1. ([106]) *Every formula of L is decidable and an equivalent quantifier-free formula can be found.*

A noticeable consequence of this is that every constraint problem can be given an answer in terms of polynomial equalities and inequalities, *i.e.* without radicals, logarithms or exponentials. Tarski went on to prove that most problems in elementary algebra and geometry can be solved by QE and gave a QE algorithm that was later found to be impractical.

Note that Tarski's decision procedure can also be extended in a rather direct manner to yield a decision procedure for the complex numbers as well. However, this problem had already been addressed earlier this century in the work of Kronecker and Hermann, among others. Their approach was later codified in the theory of resultants and resolvents - the cornerstone of *elimination theory.*

Problems of type (2) were not considered by Tarski: their roots lie in algebraic geometry and topology rather than logic. To state the main result for description problems, we need some definitions:

- a subset $X \subset E^r$ is *semi-algebraic* if it has a quantifier-free formula;
- a *region* is a non-empty connected subset of E^r;
- a *decomposition* of $X \subset E^r$ is a finite collection of disjoint regions whose union is X;
- for $0 \leq i \leq r$, a *i-cell* in E^r is a subset of E^r homeomorphic to E^i;
- given $X \subset E^r$, a decomposition of X is *cellular* if each of its regions is a cell;
- a decomposition is *algebraic* if each of its regions is a semi-algebraic set.

The definition of a semi-algebraic set being given, Tarski's theorem may be rephrased as follows: if ϕ is first-order then

$$\{(x_1, \ldots, x_n) \in \mathbb{R}^n \ / \ \phi(x_1, \ldots, x_n) \text{ is satisfied}\}$$

is a semi-algebraic subset of $\mathbb{R}^n$. Real algebraic and semi-algebraic sets are of great importance in several aspects of algorithmic problems. A first very important reason is because they are the geometric objects that are the most calculable and, to a first approximation, every geometric set in $\mathbb{R}^n$ can be assumed semi-algebraic. A second characteristic is their "closure" under projection: the projection of a semi-algebraic subset of $\mathbb{R}^{n+k}$ onto $\mathbb{R}^n$ is a semi-algebraic subset of $\mathbb{R}^n$.

Then if Set(ϕ) is the set defined by a formula ϕ, the next major step in QE is due to Collins, who gave a different and far more efficient method:

Theorem 2. ([20]) *Given a first-order formula $\phi(x_1, \ldots, x_n)$ of L, there is an algorithm that finds an algebraic cellular decomposition D of Set(ϕ) and gives, for each cell of D, its dimension, an algebraic sample point belonging to it and a quantifier-free formula defining it.*

Note that in the statement of this theorem, ϕ is not assumed to be quantifier-free. Collins' method thus yields both a QE algorithm (known as *cylindrical algebraic decomposition* - cad) and a description algorithm. Collins' work has revived interest in the feasibility of QE as a means of solving nontrivial problems in elementary algebra and geometry. It gives a particular kind of cellular decomposition in which a certain set of polynomials under consideration has constant sign in each cell. A nice introduction to cad is given in [46].

Collins's cad is a double-exponential time procedure which is still the most commonly used for practical tasks. However, the more recent results of [8, 73] have improved the known upper bounds on this problem.

Recent results have improved the computational complexity of cad algorithms. Indeed, the stratification algorithms of [19, 90], given n polynomials in D variables, determine a decomposition of $\mathbb{R}^D$ into $O(n^{2D-3+\varepsilon})$ sign-invariant connected cells of simple shape for which point-location queries can be answered in $O(\log n)$ time. This decomposition can be obtained in singly exponential time: $O(n^{2D+1})$ deterministic time and $O(n^{2D-3+\varepsilon})$ (for $D \geq 3$) or $O(n^{2+\varepsilon})$ (for $D = 2$) randomised expected time.

2.4. Mixed problems in geometric reasoning

Many important and difficult mathematical problems can be expressed in this theory (including for instance almost all motion planning problems), for example the solvability of systems of multivariate polynomial equations and of nonlinear optimisation problems. So let us give some more examples of problems that fall in this category and pursue examples introduced earlier.

2.4.1. Intersection of surfaces. Let us go back to the example given above to illustrate the query type of problems. Collins' algorithm gives a sign-invariant cad of E^3. By evaluating $\phi(x,y,z)$ at the sample points of the E^3 cells, we may learn for instance that $\mathrm{Set}(\phi)$ is non-empty (and thus that the two surfaces intersect, answering the decision problem) and that it consists in three 1-cells and one 0-cell. We infer that this intersection curve is 1-dimensional (since it contains a 1-cell). If the union of the four cells considered is connected, then we also have that the intersection curve has only one connected component. In addition, the cad provides sufficient data to be able to draw the actual picture of the intersection curve.

2.4.2. Implicitisation. Consider a parametric surface: $\mathbf{x} = (x,y,z) = (f(u,v), g(u,v), h(u,v))$. Implicitising this surface amounts to eliminating the quantifiers in the formula

$$(\exists u)(\exists v)\ [x = f(u,v) \wedge y = g(u,v) \wedge z = h(u,v)].$$

Cylindrical algebraic decomposition then allows to solve this constraint problem and to describe the equivalent quantifier-free formula. As above, the cad of E^3 gives us information about the implicit equation: number and dimensionality of its connected components among others.

2.4.3. Geometric theorem proving. When the hypotheses of a geometric problem can be set as polynomial equations relating the coordinates of the points involved, the propositions that can be deduced from the hypotheses will include everything that can be expressed as polynomials in the *ideal generated*[1] by the hypotheses. To obtain these propositions, one can set up a formula as above and eliminate the quantifiers using symbolic algebra tools (Wu's method [114] and/or Gröbner bases - see later).

A basic example of such theorem proving is as follows. Find the conditions on a and b such that the ellipse $f(x,y) = ax^2+b(y-1)^2 = 1$ lies entirely inside the ellipse $g(x,y) = 2x^2+y^2 = 2$. This sets up as:

$$(\forall x)(\forall y)\ [f(x,y) < g(x,y) \wedge a > 0 \wedge b > 0].$$

Existing algorithms are of great theoretical interest but apart from theorem proving do not have yet real practical significance. The main reason is that the size of the output may be doubly exponential, so that only simple problems can be solved.

The implementation and further improvement of QE methods is based on the efficient design of various sub-algorithms from other parts of computer algebra. For instance, a good implementation of cad relies heavily on efficient algorithms for real root isolation, computations with real algebraic numbers, computing resultants, finding the common factors of multivariate polynomials ...

The following two sections are devoted to the solutions that have been proposed for solving some of these problems, with emphasis put on the calculation of the solutions of polynomial systems. Section 3 treats methods based on resultants, whereas sections 4 and 5 considers those based on Gröbner bases respectively in the complex case and in the real case.

3. Resultant-based techniques

The constructive approach to the problem of deciding when a system of multivariate polynomial equations has a common algebraic solution has a long history. One of the principal efforts of the algebraists of the late 19^{th} century was the development of such methods. The techniques explored also allowed to construct a description of the set of common solutions of these polynomials, though better descriptions are constructible today. Resultant-based techniques formed the backbone of classical elimination theory, as presented in [58, 107]: the study of methods for solving systems of polynomials equations by repeated elimination of variables. They also played a major role in the development of invariance theory in the past century [33, 35, 39, 111].

Recent surveys of resultant-based methods can be found in [98, 103]. An overview of classical elimination methods is given in [49]. The reader will find more details in [47], where an algebraic development of the theory of classical multivariate resultants is presented, and in [56], for forming resultants of overdetermined systems of polynomial equations, *i.e.* one has more equations than unknowns.

Note that resultants have been for some time regarded as less efficient computational tools than methods based on Gröbner basis, but recent methods prove that they may still be the method of choice in some cases, as advocated in [103]. For instance, Rege [71, 72] has applied resultants to geometric theorem proving and obtained impressive speedups over Wu's method.

The following sections give an introduction first to the classical approaches to the resultant (sections 3.1 and 3.2) and then to the modern sparse resultant-based techniques (sections 3.3 and 3.4). We illustrate some of the key methods with theoretical examples.

3.1. Introduction

Definition 2. A *resultant* is an algebraic criterion for determining when a pair of univariate polynomials have a common root, expressed in terms of the coefficients of the given polynomials.

One particularly important application of resultants is in the construction of a description of the projection of an algebraic set. Such a method is useful in algorithms which solve problems by recursion on dimension. For instance, to solve a geometric reasoning problem in $p+1$ dimensions, in which a given geometric object is defined by polynomial equalities, first construct the projection of the given set onto p-dimensional space, recursively solve the problem there and lift this solution back to higher dimensional space. Note that Collins' cad makes use of this general scheme, albeit in a less direct manner.

The main drawback of this class of approaches is that the size of the description of a projection computed with the resultant can be exponential in the size of the description of the original set. Iterating p times will then result in a double-exponential blow-up in the size of the polynomials involved. On the other hand, it is able to eliminate an arbitrary subset of variables in time which is only exponential in the number of variables.

3.2. Classical resultants

Suppose given a system of m equations in n variables. Depending on the relative values of m and n, several types of resultants have been constructed. The case $m = 2$ and $n = 1$ is the traditional acception of the term resultant: it is computed as the determinant of the Sylvester matrix [105]. When $m = n+1$ $(n \geq 2)$, the resultant is computed as the quotient of two determinants and known as the *Macaulay resultant* [57]. When $m = n$ $(n \geq 2)$, we deal with the u-resultant (or with the generalised characteristic polynomial - GCP [18]). Finally, when $m < n, n \geq 2$, what is usually computed is a Macaulay resultant using $m-1$ variables, treating the remaining variables as coefficients.

Note that many extensions of classical resultants have been considered over the years. We only present here some of the most interesting techniques. The reader is invited to consult other sources like [54] for more.

We sketch in what follows the construction of the Sylvester resultant and give a presentation of the general approach to the resultant, which is the basis of the Macaulay resultant and the u-resultant that we explain next.

3.2.1. Sylvester resultant. Suppose given the following two univariate polynomials:

$$\begin{cases} f(x) = a_r x^r + \cdots + a_0, \\ g(x) = b_s x^s + \cdots + b_0. \end{cases}$$

Definition 3. The *Sylvester resultant* of f and g is the determinant of the $(r+s)\times(r+s)$ Sylvester matrix:

$$R_{r,s}(f,g) = \begin{vmatrix} a_0 & a_1 & \cdots & a_r & 0 & 0 & \cdots & 0 \\ 0 & a_0 & \cdots & a_{r-1} & a_r & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_0 & a_1 & a_2 & \cdots & a_r \\ b_0 & b_1 & \cdots & b_{s-1} & b_s & 0 & \cdots & 0 \\ 0 & b_0 & \cdots & b_{s-2} & b_{s-1} & b_s & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_0 & b_1 & b_2 & \cdots & b_s \end{vmatrix}.$$

We then have the property that $R_{r,s}(f,g)$ is zero if and only if f and g have a root in common.

The Sylvester resultant may be expressed in a different way, using a method of Bezout. Suppose that $a_0 = 1$, and consider the division of g by f:

$$\frac{g(x)}{f(x)} = r_0 + r_1 x + r_2 x^2 + r_3 x^3 + \cdots$$

Then the resultant is the $r \times r$ determinant:

$$R_{r,s}(f,g) = \begin{vmatrix} r_s & \cdots & r_{s+r-1} \\ \vdots & \ddots & \vdots \\ r_{s-r+1} & \cdots & r_s \end{vmatrix}.$$

Finally, if $\Delta(f)$ denotes the discriminant of f, which may be expressed in terms of resultants as follows:

$$\Delta(f) = \frac{1}{a_r} R_{r,r-1}(f,f'),$$

then the Sylvester resultant of f and g is:

$$(R_{r,s}(f,g))^2 = (-1)^{rs} \frac{\Delta(fg)}{\Delta(f)\Delta(g)}.$$

Example 4: If $f(x) = 6x^2 - 7x + 1$ and $g(x) = x^2 + 2x - 3$, then

$$\frac{g(x)}{f(x)} = -3 - 19x - 114x^2 - 684x^3 + \cdots$$

In turn, using Bezout's formulation,

$$R_{2,2}(f,g) = \begin{vmatrix} -114 & -684 \\ -19 & -114 \end{vmatrix} = 0,$$

which is correct since $x = 1$ is a common root of f and g. □

Finally, call $R^{(k)}_{r,s}(f,g)$ the determinant of the $(r+s-2k) \times (r+s-2k)$ sub-matrix of the Sylvester matrix obtained by deleting the first k rows, the last k rows, the first k columns and the last k columns. Then f and g have exactly k roots in common if $R_{r,s}(f,g) = 0, R^{(1)}_{r,s}(f,g) = 0, \ldots, R^{(k-1)}_{r,s}(f,g) = 0$ and $R^{(k)}_{r,s}(f,g) \neq 0$.

Formulas of the Sylvester type (*i.e.* determinants of a single matrix) are the best one can hope for a resultant. But though they are available in a number of important cases, most of the time it is not possible to express the resultant in the form of a single determinant, because doing so would for instance introduce an extraneous factor which may be hard to separate from the resultant itself. Sometimes a criterion for common roots may be written as the ratio of two determinants, as in the Macaulay resultant. The state of the art on this issue can be found in [113]. A formulation related to the Sylvester resultant is the Dixon resultant [24], for the case of three polynomials in two variables.

3.2.2. Resultant of a multivariate polynomial system. When dealing with more than one variable, we shall be speaking of a multivariate resultant or simply resultant for short. Let $f_1, \ldots, f_n$ be general forms of degrees $d_1, \ldots, d_n$ and let

$$d = 1 + \sum_{i=1}^{m} (d_i - 1).$$

Suppose we abbreviate a monomial $x_1^{a_1} \cdots x_n^{a_n}$ by x^a and write $|a| = a_1 + \cdots + a_n$ for its degree. Define S_i to be

$$S_i = \{x^a \ / \ |a| = d \text{ and } i \text{ is the smallest index such that } x_i^{d_i} \text{ divides } x^a\}.$$

From this, we can form the following system of $\binom{d+n-1}{d}$ equations:

$$\begin{cases} \frac{x^a}{x_1^{d_1}} & f_1 & = & 0 & \text{for all } x^a \in S_1, \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{x^a}{x_n^{d_n}} & f_n & = & 0 & \text{for all } x^a \in S_n. \end{cases}$$

Each of these equations can be written as a linear combination of the $\binom{d+n-1}{d}$ monomials of degree d. The coefficients of these linear combinations form a square matrix Δ_n whose determinant is denoted by D_n.

The above construction may be carried out with any variable as the last variable. For all i choose an ordering of the variables in which x_i is the last one and call D_i the corresponding determinant. Then it may be proved that the resultant of the initial system is the greatest common divisor of $D_1, \ldots, D_n$ (a theorem of Hurwitz [111]). This result was used by Macaulay to prove that D_n divided by the resultant is the determinant of a certain explicit sub-matrix of Δ_n [57]. The u-resultant also uses this general scheme.

3.2.3. Macaulay resultant. When $m - n = 1$, the Macaulay resultant tests whether or not a common solution exists. So let $f_1, \ldots, f_m$ be a set of inhomogeneous equations in $m-1$ variables. First homogenise the system and compute its overall degree d. Assume that the homogeneous variables are $x_i, i = 1, \ldots, m$. Associate to equation f_i the variable x_i and consider the monomials of degree d involving the m (homogeneous) variables. Call such a monomial reduced if there is only one variable x_i for which the exponent of x_i in the monomial is larger than or equal to d_i, the degree of f_i (the variable is called *big*).

Definition 4. The *Macaulay resultant* is formed as a ratio of determinants:

$$R = \frac{\|A\|}{\|M\|}.$$

The construction of the matrix A of size $\binom{n+d}{d}$ goes as follows. Think of each row and column of A as being labelled by one monomial of degree d. Suppose fixed a column of A (and thus a monomial ω). Find the first big variable in ω and call it x_j. Construct the polynomial

$$\frac{f_j.\omega}{x_j^{d_j}}.$$

The elements of the column considered are then the coefficients of this polynomial. The M matrix is then a sub-matrix of A consisting of the elements having row and column monomial labels which are not reduced.

The original system has solutions if the determinant of A is zero while that of M is nonzero. It may sometimes happen that both determinants vanish while the system still admits solutions. This may be circumvented by using symbolic coefficients (often only one symbolic coefficient is enough) and replacing by the numerical values at the end.

Example 5: Consider the following polynomial system and its homogenisation:

$$\left\{\begin{array}{l} f_1 = \lambda x_1 + x_2 + 1 \\ f_2 = x_1^2 + x_2^2 - 2 \\ f_3 = 3x_1^2 - x_2 - 2 \end{array}\right. \Rightarrow \left\{\begin{array}{l} f_1 = \lambda x_1 + x_2 + x_3 \\ f_2 = x_1^2 + x_2^2 - 2x_3^2 \\ f_3 = 3x_1^2 - x_2 x_3 - 2x_3^2 \end{array}\right. .$$

We now follow Definition 4 for constructing the Macaulay resultant of this system. The overall degree is $d = 3$ and A is a 10×10 matrix. Consider the monomial $x_1^2 x_2$ of degree 3. The first big variable of this monomial is x_1 (since the exponent of x_1 is larger than or equal to d_1). The associated polynomial is:

$$\frac{f_1 x_1^2 x_2}{x_1} = \lambda x_1^2 x_2 + x_1 x_2^2 + x_1 x_2 x_3.$$

We can therefore fill the column labelled $x_1^2 x_2$. The element of the row labelled $x_1^2 x_2$ is λ, the element of the row labelled $x_1 x_2^2$ is 1, ... The A matrix is as follows:

	x_1^3	$x_1^2 x_2$	$x_1^2 x_3$	$x_1 x_2^2$	$x_1 x_2 x_3$	$x_1 x_3^2$	x_2^3	$x_2^2 x_3$	$x_2 x_3^2$	x_3^3
x_1^3	λ	0	0	0	0	0	0	0	0	0
$x_1^2 x_2$	1	λ	0	0	0	0	1	0	3	0
$x_1^2 x_3$	1	0	λ	0	0	0	0	1	0	3
$x_1 x_2^2$	0	1	0	λ	0	0	0	0	0	0
$x_1 x_2 x_3$	0	1	1	0	λ	0	0	0	0	0
$x_1 x_3^2$	0	0	1	0	0	λ	0	0	0	0
x_2^3	0	0	0	1	0	0	1	0	0	0
$x_2^2 x_3$	0	0	0	1	1	0	0	1	-1	0
$x_2 x_3^2$	0	0	0	0	1	1	-2	0	-2	-1
x_3^3	0	0	0	0	0	1	0	-2	0	-2

The non-reduced monomials of degree 3 are $x_1 x_2^2, x_1 x_3^2$, so matrix M is:

	$x_1 x_2^2$	$x_1 x_3^2$
$x_1 x_2^2$	λ	0
$x_1 x_3^2$	0	λ

The resultant in this case is found to be:

$$R = (\lambda - 2)(\lambda + 2)(\sqrt{2}\lambda - 1)(\sqrt{2}\lambda + 1).$$

$\lambda = -2$ yields the solution $\{x_1 = 1, x_2 = 1\}$, $\lambda = 2$ the solution $\{x_1 = -1, x_2 = 1\}$, $\lambda = \frac{1}{\sqrt{2}}$ the solution $\{x_1 = \frac{\sqrt{2}}{3}, x_2 = -\frac{4}{3}\}$ and $\lambda = -\frac{1}{\sqrt{2}}$ the solution $\{x_1 = -\frac{\sqrt{2}}{3}, x_2 = -\frac{4}{3}\}$. Note that $\lambda = 0$ would have made the determinant of M to vanish, but not that of A. □

3.2.4. u-resultant. When $m = n$ and the equations are inhomogeneous, the equations can usually be solved, after homogenisation, by adding the u-equation and forming the Macaulay resultant. In some cases, even though the original system has a zero-dimensional solution set, adding the u-equation can give birth for instance to a zero-set consisting of a one-dimensional component (called a component of *excess dimension*) and of isolated points. In this case, the component of excess dimension masks the desired solutions and it is better to use the GCP approach.

Definition 5. Let $f_1, \ldots, f_m$ be a set of m inhomogeneous equations in m variables. The *u-equation* is simply a hyperplane

$$u_1 x_1 + \cdots + u_{m+1} x_{m+1} = 0,$$

where the u_i are symbolic coefficients and x_{m+1} is the homogenising variable. The *u-resultant* R is then the Macaulay resultant of the homogeneous f_i and the u-equation.

Once R is known, it is factored into linear factors. Given such a factor L, the coordinates of each solution of the original system are given as ratios

$$x_i = \frac{\text{coefficient of } u_i \text{ in } L}{\text{coefficient of } u_{m+1} \text{ in } L}.$$

Example 6: Consider the following system:

$$\begin{cases} f_1 = -2x_1 + x_2 + 1 \\ f_2 = 3x_1^2 - x_2 - 2 \end{cases} \Longrightarrow \begin{cases} f_1 = -2x_1 + x_2 + x_3 \\ f_2 = 3x_1^2 - x_2 x_3 - 2x_3^2 \end{cases}.$$

We add the u-equation

$$f_3 = u_1 x_1 + u_2 x_2 + u_3 x_3.$$

The overall degree is $d = 2$ and A is a 6×6 matrix. Thus according to Definitions 4 and 5, A is computed as above:

	x_1^2	x_1x_2	x_1x_3	x_2^2	x_2x_3	x_3^2
x_1^2	-2	0	0	3	0	0
x_1x_2	1	-2	0	0	u_1	0
x_1x_3	1	0	-2	0	0	u_1
x_2^2	0	1	0	0	u_2	0
x_2x_3	0	1	1	-1	u_3	u_2
x_3^2	0	0	1	-2	0	u_3

The M matrix is reduced to the element -2. The u-resultant is:

$$R = \frac{1}{2}(u_1 + 5u_2 - 3u_3)(u_1 + u_2 + u_3).$$

The first linear factor yields the solution $\{x_1 = -\frac{1}{3}, x_2 = -\frac{5}{3}\}$ and the second the solution $\{x_1 = 1, x_2 = 1\}$. □

When the u-resultant vanishes or results in an indeterminacy (0/0), even using symbolic coefficients, this means that a component of excess dimension exists. In this case, the GCP is computed as follows:

$$R = \frac{|A - sI|}{|M - sI|},$$

where I is the identity matrix and s is a perturbation parameter. While computing each determinant, we keep the lowest remaining power of s to do the rest.

3.3. Sparse resultants

It is clear that in most cases the constructions described above, for instance the Macaulay resultant, will result in sparse matrices (*i.e.* matrices having most of their entries equal to zero). And thus much of the computation time will be wasted evaluating expressions that will eventually vanish. Another - related - motivation for sparse resultants comes from the following example.

Example 7: Consider the following three equations [103]:

$$\begin{cases} f = a_0 x + a_1 y + a_2 xy, \\ g = b_0 + b_1 xy + b_2 y^2, \\ h = c_0 + c_1 xy + c_2 x^2. \end{cases} \quad (1)$$

Forming the Macaulay resultant yields a determinant which vanishes identically. Indeed, $(1, 0, 0)$ is always solution of the homogenised system and the three quadrics always intersect in $\mathbb{P}^2$. But the resultant does not give any information as when the quadrics intersect in the affine plane $\mathbb{C}^2$. For this, a new resultant has to be formed, the sparse resultant. □

A good introduction to sparse resultants and applications can be found in [26, 99]. Formalisation of the sparse resultant is recent and due to Gelfand, Kapranov and Zelivinsky [30], jointly with Sturmfels [104] for arbitrary systems.

We shall here describe one of the methods that have been proposed for computing sparse resultants. But before that, we shall present the notions on which this method is based, that of Newton polytopes and mixed subdivisions.

3.3.1. New polytopes. Suppose given a polynomial f in the variables $x_1, \ldots, x_n$ written as follows

$$f = \sum_{\alpha=(\alpha_1,\ldots,\alpha_n)\in\mathbb{Z}_+^n} c_\alpha x_1^{\alpha_1}\cdots x_n^{\alpha_n}.$$

Then the *Newton polytope* of f is defined to be the convex hull in $\mathbb{R}^n$ of the integral lattice points $\alpha \in \mathbb{R}_+^n$ for which $c_\alpha \neq 0$.

The Minkowski sum of two convex polytopes P_1 and P_2 of $\mathbb{R}^n$ is a convex polytope defined as follows:

$$P = P_1 + P_2 = \{p_1 + p_2 \ / \ p_1 \in P_1 \text{ and } p_2 \in P_2\}.$$

Denote by $\mathrm{Vol}(P)$ the usual volume of P in $\mathbb{R}^n$. Given convex polytopes $P_1, \ldots, P_n$, the *mixed volume* $\mathrm{Mv}(P_1, \ldots, P_n)$ of the collection is the coefficient of $\lambda_1 \cdots \lambda_n$ in the expansion of $\mathrm{Vol}(\lambda_1 P_1 + \cdots + \lambda_n P_n)$ seen as a polynomial in $\lambda_1, \ldots, \lambda_n$.

Theorem 3. ([11]) *Suppose $f_1, \ldots, f_n$ are n equations in n variables. The number of common zeros in the algebraic torus (*i.e. *$(\mathbb{C}^*)^n$, with $\mathbb{C}^* = \mathbb{C}\setminus\{0\}$) is either infinite or is less than the mixed volume $\mathrm{Mv}(P_{f_1}, \ldots, P_{f_n})$ of the Newton polytopes P_{f_i} associated to the equations f_i.*

This bound, known as the *BKK bound* (for Bernstein - Kushnirenko - Khovanskii), is at most as high as Bezout's bound, the product of the total degrees, and is usually significantly smaller for systems of practical interest. More on the complexity of real algebraic sets in connection to Newton polyhedra can be found in [9], improving the famous Thom - Milnor bound [61].

3.3.2. Defining the sparse resultant. For the content of this section, we follow [103]. Let $\mathcal{A}_0, \ldots, \mathcal{A}_n$ be finite subsets of $\mathbb{Z}^n$. Consider a system of $n+1$ polynomials in n variables $x = (x_1, \ldots, x_n)$ of the following form

$$f_i(x) = \sum_{a\in\mathcal{A}_i} c_{ia} x^a, \qquad i = 0, 1, \ldots, n, \qquad (2)$$

with the notation $x^a = x_1^{a_1}\cdots x_n^{a_n}$ for $a = (a_1, \ldots, a_n) \subset \mathbb{Z}^n$. The set $\mathcal{A}_i$ is called the *support* of the polynomial f_i. Let m_i be its cardinal. For instance, the support of f in system 1 is $\{(1,0),(0,1),(1,1)\}$.

The vector of all coefficients c_{ia} in 2 defines a point in the product of complex projective spaces $\mathbb{P}^{m_0-1}\times\cdots\times\mathbb{P}^{m_n-1}$. Denote by Z the subset of those systems 2 which have a solution in $(\mathbb{C}^*)^n$. Let $\bar{Z}$ be the closure of Z in $\mathbb{P}^{m_0-1}\times\cdots\times\mathbb{P}^{m_n-1}$. The variety $\bar{Z}$ turns out to be irreducible and defined over $\mathbb{Q}$. Then:

Definition 6. If the codimension of $\bar{z}$ is 1, the *sparse resultant* $\mathcal{R}$ is defined as the unique (up to sign) irreducible polynomial in $\mathbb{Z}[\cdots, c_{ia}, \cdots]$ which vanishes on the hypersurface $\bar{Z}$. If the codimension of $\bar{Z}$ is larger than 1, $\mathcal{R}$ is defined to be the constant 1.

Using the BKK bound, one can then define the following result if the set $\{\mathcal{A}_0, \ldots, \mathcal{A}_n\}$ satisfies the property of being essential (see [103]):

Theorem 4. *Let Q_i be the convex hull of $\mathcal{A}_i$. For all $i \in \{0, \ldots, n\}$ the degree of $\mathcal{R}$ in the i-th group of variables $\{c_{ia}, a \in \mathcal{A}_i\}$ is a positive integer, equal to the mixed volume*

$$\mathcal{M}(Q_0, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n) = \sum_J (-1)^{card(J)}\, Vol\left(\sum_{j\in J} Q_j\right),$$

where J ranges over all subsets of $\{0, \ldots, i-1, i+1, \ldots, n\}$.

Example 8: With system 1, the three Newton polytopes are triangles (see Fig. 1[2]). The mixed volume of two polygons in the plane is equal to the area of their Minkowski sum, minus their respective areas. Here, the areas of Q_1, Q_2, Q_3 are respectively $1/2$, 1 and 1. The area of the Minkowski sum $Q_2 + Q_3$ is 4 and the areas of $Q_1 + Q_3$ and $Q_1 + Q_2$ (see Fig. 1.c) are equal to $9/2$. Thus, we have:

$$\begin{cases} \mathrm{Mv}(Q_1, Q_2) = \mathrm{Mv}(Q_1, Q_3) = 3, \\ \mathrm{Mv}(Q_2, Q_3) = 2. \end{cases}$$

This means by Theorem 4 that the sparse resultant should be quadratic in (a_0, a_1, a_2) and cubic in (b_0, b_1, b_2) and (c_0, c_1, c_2). □

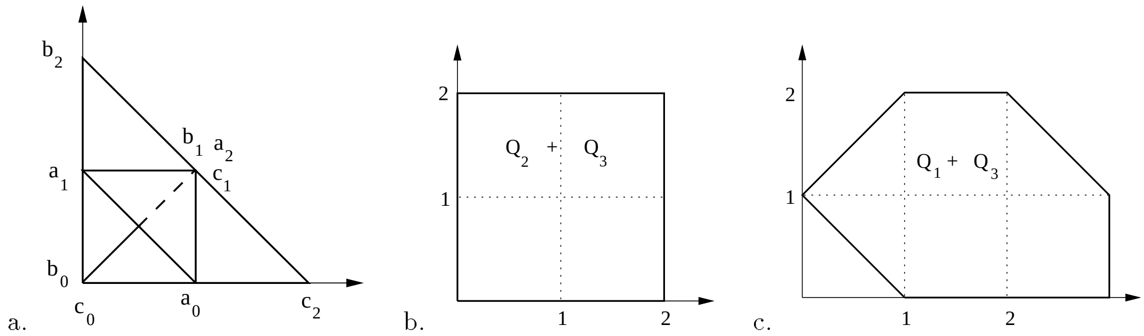

Fig. 1. Examples of polytopes and Minkowski sums. a. The three triangles of system 1. b. The sum $Q_2 + Q_3$. c. The sum $Q_1 + Q_3$.

3.3.3. Computing the sparse resultant. For constructing sparse resultants, several different methods are known. In [17, 26], the sparse resultant is expressed through various determinants in the original coefficients. We give a brief description of this method here, which produces a matrix B whose determinant is a non-zero multiple of the resultant needed.

A *mixed subdivision* Δ of a polytope $Q = Q_1 + \cdots + Q_m \subset \mathbb{R}^n$ is a polyhedral subdivision [26] of Q such that every polyhedron $F \in \Delta$ is of the

How then do we compute the sparse resultant? As we have already mentioned, it is hopeless in the general case to find a matrix whose determinant is the resultant. However, the sparse resultant divides the determinant of some matrix B and we now give a hint at how this matrix may be constructed. Let Q be the Minkowski sum of the Newton polytopes Q_{f_i} of $f_i, i = 1, \ldots, n+1$. The rows and columns of B are essentially indexed by the integer lattice points in the Minkowski sum Q. The specific entries m_{ij} of B are determined using a mixed subdivision of Q.

Explicitly, the Minkowski sum Q is slightly perturbed (as described in [25]) so that each integer lattice point will lie in the interior of a polyhedron F in the mixed subdivision Δ. A difficulty overcome by the Canny-Emiris method is to ensure that the determinant of a matrix set up this way does not vanish identically. This is achieved by forcing the leading monomial (with respect to some term order) to be non-zero by making it appear on the diagonal of matrix B.

Example 8 continued: The mixed subdivision rules in the case of system 1 produce the fol-

form $F = F_1 + \cdots + F_m$, where F_i is a face of Q_i and dim $F = \sum_{i=1}^m$ dim F_i. A *mixed facet* $F \in \Delta$ is a polyhedron such that dim $F = n$ and every F_i has dimension ≤ 1. With the above definitions, it turns out that given a mixed subdivision of Q:

$$\mathrm{Mv}(Q_1, \ldots, Q_m) = \sum_{\text{mixed facets } F \in \Delta} \mathrm{Vol}(F).$$

Fast algorithms are known for finding mixed subdivisions [26].

lowing 10×10 matrix:

$$B = \begin{pmatrix} a_1 & 0 & 0 & 0 & 0 & a_2 & 0 & 0 & 0 & a_0 \\ 0 & a_1 & a_2 & 0 & 0 & a_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_1 & 0 & 0 & 0 & a_0 & a_2 & 0 & 0 \\ b_0 & 0 & b_1 & b_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & b_2 & b_0 & 0 & b_1 & 0 & 0 \\ 0 & b_2 & 0 & 0 & 0 & b_1 & 0 & 0 & b_0 & 0 \\ 0 & 0 & b_2 & 0 & 0 & 0 & b_1 & 0 & 0 & b_0 \\ 0 & 0 & 0 & 0 & c_2 & c_0 & 0 & c_1 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & c_1 & 0 & 0 & c_0 & 0 \\ 0 & 0 & c_2 & 0 & 0 & 0 & c_1 & 0 & 0 & c_0 \end{pmatrix}.$$

The sparse resultant in this case has degree $8 = \mathrm{Mv}(Q_1, Q_2) + \mathrm{Mv}(Q_1, Q_3) + \mathrm{Mv}(Q_2, Q_3)$ and the determinant of B is $a_1 b_2$ times this resultant. □

Example 9: Consider the following system of equations (see [26]):

$$\begin{cases} f_1 = a_{11} + a_{12}xy + a_{13}x^2y + a_{14}x, \\ f_2 = a_{21}y + a_{22}x^2y^2 + a_{23}x^2y + a_{24}x, \\ f_3 = a_{31} + a_{32}y + a_{33}xy + a_{34}x. \end{cases}$$

The Newton polytopes associated to these polynomials are displayed in Figure 2.

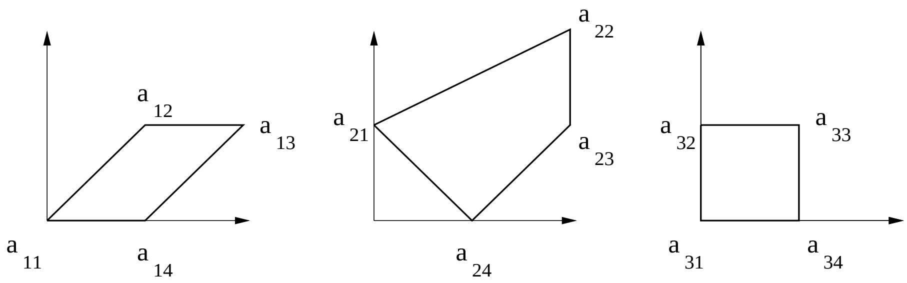

Fig. 2. A set of Newton polytopes.

The three pairs have the following mixed volumes: $\mathrm{Mv}(Q_1, Q_2) = 4$, $\mathrm{Mv}(Q_1, Q_3) = 3$ and $\mathrm{Mv}(Q_2, Q_3) = 4$, while the respective Bezout bounds are 12, 6 and 8. This means that the sparse resultant has degree $4+3+4 = 11$, while the classical resultant in this case would have degree $12 + 6 + 8 = 26$.

The slightly perturbed subdivision of the Minkowski sum Q is illustrated on Figure 3. Each cell is labelled with the indices of the Newton polytope vertices that contribute to it. The B matrix for this example is given in [26, page 43]. □

3.4. Sparse systems and root counting

Sparsity of polynomials systems and the counting of (affine) roots of such systems are topics that are

M. Rojas [77, 80, 81] has presented many extensions to the BKK bound which counts isolated solutions in the algebraic torus (*i.e.*, $(\mathbb{C}^*)^n$). It is known that if we assume fixed the monomial term structure, then the BKK bound fails to be an exact root count only on a codimension 1 algebraic subset of the coefficient space. Rojas has first given a method, based on the sparse resultant, for counting isolated roots in the algebraic torus that fails on a codimension ≥ 2 subset of the coefficient space and has recently developed a convex geometric which always work. In addition, the author presents classifications of the subsets of the coefficients of the defining equations whose genericity guarantees exactness in the generalised bounds.

currently being largely investigated [25, 42, 81]. In particular, the results presented above have already been extended in several different ways, which is the subject of this closing section on resultants. Root counting is important because the complexity of isolating, describing and computing the roots of a system is a function of the number of roots.

First note that algorithms are now known for computing classical resultants by taking into account the sparsity of the input system. For instance, Emiris has developed a sparse version of the u-resultant [26]. Recently, Rojas has presented a method for computing the GCP of sparse systems [79].

Optimal convex geometric upper bounds are also known on the number of roots in all of $\mathbb{C}^n$ and in subregions other than $(\mathbb{C}^*)^n$ [43, 55, 81].

No geometric formula is however known for the maximal (asymptotic) number of real roots of a polynomial system, where the coefficients of the polynomial equations are allowed to vary in some given space, though some conjectures have been formulated [102] (simplified by Itenberg and Roy [44]). An explicit formula giving the expected number of real roots of certain random sparse polynomial systems has been given by Rojas [78].

Example 10: Consider the following system of polynomial equations [80]:

$$\begin{aligned} f_1 &= \alpha_1 y^2 + \alpha_2 y^4 + \alpha_3 x y^5 + \alpha_4 x^2 y^5 \\ &\qquad + \alpha_5 x^2 y^7 + \alpha_6 x^4 y^8, \\ f_2 &= \beta_1 x + \beta_2 x^2 + \beta_3 x y + \beta_4 x^2 y + \beta_5 x^6 y^2 \\ &\qquad + \beta_6 x^6 y^3 + \beta_7 x^7 y^3 + \beta_8 x^9 y^5. \end{aligned}$$

The BKK bound (computed using the mixed volume code of I. Emiris [25]) says that there are at most 32 isolated roots in $(\mathbb{C}^*)^2$. The classical Bezout bound says that there are no more than 168 isolated affine roots of this system. The best generalised Bezout bound gives an upper bound of

$$(\deg_x f_1)(\deg_y f_2) + (\deg_x f_2)(\deg_y f_1) = 92$$

for the number of such solutions. The Affine Point Theorem I [81] yields an upper bound of 53, while a recent result [80] gives the optimal upper bound of 38. The author also proves that the system has 38 affine isolated roots (counting multiplicities) generically, iff

$$(\alpha_5\beta_8 - \alpha_6\beta_3)\alpha_1\alpha_2\beta_2\beta_3\beta_8 \neq 0. \qquad \square$$

4. Symbolic system solving in the complex case

When dealing with a polynomial system, what computer algebra has to offer is to construct a new system of equations with the same solutions as the initial one, but with a "simpler" structure. What simpler means is that the amount of computations to determine the desired information on the solutions is smaller.

This leads to the notion of a Gröbner basis which we shall introduce and develop in this section. The main properties of Gröbner bases are presented along with some methods to turn them into algorithms. In particular, it is shown how to reduce the problem of solving a polynomial system to a linear algebra problem. Interested readers will find more on these subjects in the recent surveys [31, 110]. Introductions to Gröbner bases can be found in [1] and [36] (with a particular emphasis on applications). Note that while computations on resultants can be performed using floating point arithmetic with the major drawback that they can only be used in generic situations (some people are attacking this problem - see [64], Gröbner bases computations require exact arithmetic (except when the problem can ultimately be transformed into a linear algebra one) but do not fail in non-generic situations.

In the following sections, we present some of the properties that make Gröbner bases attractive for symbolic computations. We start with the univariate case which allows to intuitively understand what a Gröbner basis really is (section 4.1). We then show in section 4.2 how the knowledge of the Gröbner basis associated to a polynomial system gives information on the number of solutions of the system. The last section (4.3) is devoted to the results that are known for reducing the problem of solving a polynomial system, via standard bases, to a linear algebra problem.

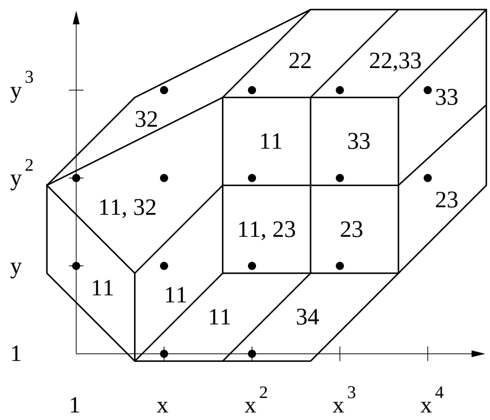

Fig. 3. The subdivision of the Minkowski sum Q.

4.1. *Gröbner bases method as generalisation of Euclid's method*

Let F be a finite collection of univariate polynomials with rational coefficients. To determine the structure of the zero set of F, a version of Euclid's algorithm [22] is applied. The basic step is a reduction: given f of highest degree monomial ax^m and g of highest degree bx^n, $m > n$, the reduction of f by g is the replacement of f by $f_1 = a/bx^{m-n}g$, where obviously the degree of f_1 is less than that of f. This process can be applied iteratively to the collection F, by choosing at each step the polynomial g of minimal degree and replacing all other polynomials by their reduction by g. Discarding all zero polynomials, we are left after a finite number of steps with a unique polynomial which is the gcd of all polynomials in F.

Both the Gröbner bases method, on which we shall focus now, and Wu's method, that we have already mentioned, are generalisations of Euclid's algorithm to the multivariate case. Wu's method is particularly well-suited for geometric theorem proving. More on the differences between these two generalisations can be found in [110].

4.2. Gröbner bases: main results

Definition 7. If $\{f_i\}_{i=1,\ldots,m} \in k[x_1,\ldots,x_n]$ (k is a fixed algebraically closed field) is a basis for the ideal I (I is roughly the "span" of $\{f_i\}_{i=1,\ldots,m}$), then a *Gröbner basis* (or *standard basis*) of I is another basis $\{g_j\}_{j=1,\ldots,r}$ which has a number of properties that makes it a very powerful tool for effective methods in algebraic geometry and real algebraic geometry.

Gröbner bases were developed by Bruno Buchberger [14] (who named them after his research advisor) and have been extensively studied and developed since then. For sake of simplicity, one may assume that the basis $\{g_j\}$ is a certain completion of the original basis with respect to one of many orderings of the monomials. We shall here meet the lexicographic order (`lex`), where $a >_{\mathtt{lex}} b$ if and only if in the vector difference $a-b$ the left-most nonzero entry is positive, and the graded reverse lexicographic order (`grevlex`), where $a >_{\mathtt{grevlex}} b$ iff either $\sum a_i > \sum b_i$ or $\sum a_i = \sum b_i$ and the right-most nonzero entry in $a-b$ is negative.

We now review some of the properties of standard basis, first in the general case, and then when the set of solutions of the polynomial system considered is zero-dimensional.

4.2.1. First properties. The main property is that the polynomial system built on any Gröbner basis of a set S of polynomial equations has the same solutions as S. The problem of deciding whether a solution exists is tackled using a result which is a consequence of a theorem known as Hilbert's Nullstellensatz, a German word formed from the three simpler words: Null (= Zero), Stellen (= Places) and Satz (= Theorem).

Theorem 5. ([40]) *A set S of polynomial equations, with generated ideal I, has no solution if and only if $1 \in I$.*

Now, the Gröbner bases of I are also such that no solution exists if and only if $1 \in \{g_j\}$. So once this basis is known, one can immediately decide whether a solution exists from the set of polynomials $\{g_j\}$.

Example 11: Consider the following system of equations:

$$\begin{cases} x^2+y^2-1=0, \\ y^2-1=0, \\ xy+c=0, \end{cases}$$

where c is a constant. If $c=1$, then the Gröbner basis using **`grevlex`** is $\{1\}$ so the system has no solution by Theorem 5, while for $c=0$, the basis is $\{x, y^2-1\}$, with the solutions $(0,1)$ and $(0,-1)$. □

A Gröbner basis algorithm has the drawback of solving a more complex class of problems than the simple decision problem we were after. Indeed, this algorithm allows to solve the ideal membership problem (which is considerably harder), *i.e.* to decide whether an arbitrary polynomial is in the ideal generated by $f_1,\ldots,f_m$. Since ideal membership has been shown to be hard for exponential space, one might expect the decision problem to require no more than exponential time.

Theorem 6. *Let G be a standard basis for system S. Then S has a finite number of solutions if and only if for every unknown x_i, G has a polynomial having as leading term with respect to the ordering used (biggest monomial) a term of the form cx_i^s, $s>0$.*

Example 12: For the system,

$$\begin{cases} x^2+xy-2y^2=0, \\ x^2+x-xy-y=0, \end{cases}$$

the standard basis is $G=\{2x^2-2y^2+x-y, 2xy-2y^2-x+y\}$. Thus Theorem 5 says this system has solutions. The leading monomials of G are $2x^2$ and $2xy$. Thus y^2 does not appear as leading term in a polynomial of G and the system has infinitely many solutions (the line $x=y$) by Theorem 6. □

In the course of a computation, a Gröbner basis solver should look for factorisation in its intermediate polynomials. Indeed, if a new polynomial h appears which factors into h_1h_2, then h vanishes if either h_1 or h_2 vanishes. Thus a zero of G is either a zero of $G_1 = G \cup \{h_1\}$ or of $G_2 = G \cup \{h_2\}$.

The standard basis for G_1 and G_2 can be computed separately and such decompositions make the algorithm more efficient.

4.2.2. 0-dimensional and triangular systems. For 0-dimensional systems, Lazard [54] introduced the notion of *triangular systems*.

Definition 8. A set of polynomials $\{f_1(x_1), f_2(x_1, x_2), \ldots, f_n(x_1, \ldots, x_n)\}$ is *triangular* if for every integer k, $f_k(x_1, \ldots, x_k)$ has an invertible leading coefficient (regarded as a polynomial in x_k).

A nice result then is that every zero-dimensional system may be decomposed into triangular pieces without factorisation (only gcd computations). Also, the Gröbner basis of a 0-dimensional system with respect to $>_{\texttt{lex}}$ is triangular.

Theorem 7. (Shape Lemma) *Under some suitable assumptions (verified in most cases), the Gröbner basis with respect to $>_{\texttt{lex}}$ of a polynomial system with a finite number of solutions has the following form:*

$$x_1 - g_1(x_n), x_2 - g_2(x_n), \ldots, x_{n-1} - g_{n-1}(x_n), g_n(x_n),$$

every g_i being a univariate polynomial.

How attractive Theorem 7 may be, computing the Gröbner basis with respect to $>_{\texttt{lex}}$ may not always prove adequate because of a large computing time or because the size of the output is prohibitive.

Example 13: As an example of the result stated in Theorem 7, the standard basis of the system (with respect to $>_{\texttt{lex}}$)

$$\begin{cases} x + y^2 - z^2 + 1 = 0, \\ x - y + z^2 = 0, \\ y - z = 0 \end{cases}$$

is $\{x+1, y-z, z^2-z-1\}$. Thus $g_1(z) = 1, g_2(z) = z$ and $g_3(z) = z^2 - z - 1$. □

4.3. Gröbner bases and linear algebra

The main properties of Gröbner bases described above may not easily be turned into algorithms. One of the main difficulties lies in the fact that a stable problem may be transformed into an unstable one, and if a numerical algorithm is used to solve the transformed system (for instance the univariate problem produced by the Shape Lemma), then this may lead to numerical imprecisions. To avoid this, use is made of linear algebra methods to reduce the resolution of a polynomial system, via Gröbner bases, to a linear algebra problem. As an aid to the reader, we have written a number of Maple procedures based on the `grobner` and `linalg` packages to test the linear algebra results below[3].

Definition 9. Assume the system $f_1, \ldots, f_n$ has a finite number of solutions and has exactly n unknowns. Given a standard basis G with respect to $>$ of $f_1, \ldots, f_n$, the *normal form* of a polynomial f is a polynomial obtained after a finite number of reductions such that it no longer contains a term that is divisible by leading terms of polynomials of G.

Now, one can assert if a polynomial is a linear combination of $f_1, \ldots, f_n$ by computing its normal form with respect to the standard basis of the system:

Theorem 8. *A polynomial f can be written as a linear combination of $f_1, \ldots, f_n$ if and only if its normal form with respect to G and $>$ is 0.*

Example 14: Suppose that we use the lexicographical ordering. Let $\{f_1, f_2\} = \{x_1^2 x_2 - x_1, x_1 x_2^2 + x_1 x_2\}$. The standard basis is:

$$G = \{g_1, g_2\} = \{x_1 + x_1^2, x_1 + x_1 x_2\}.$$

Let h be the polynomial $x_1^2 x_2^2 + x_2^2$. Here is one possible way of getting a normal form of h with respect to $\{g_1, g_2\}$, through the following sequence of reductions:

$$\begin{aligned} h \xrightarrow{g_2} h_1 &= h - x_1 x_2 g_2 = x_2^2 - x_1^2 x_2 \\ \xrightarrow{g_2} h_2 &= h_1 + x_1 g_2 = x_2^2 + x_1^2 \\ \xrightarrow{g_1} h_3 &= h_2 - g_1 = x_2^2 - x_1. \end{aligned}$$

Thus the normal form of h with respect to G is $x_2^2 - x_1$ and Theorem 8 says that h is not a linear combination of f_1 and f_2. □

In the normal forms of any polynomial with respect to G and $>$, only those monomials whose exponent vector appears below the exponent vectors of the leading terms of the polynomials in G can be present. Let us denote this set of monomial by $\mathcal{A}$.

Theorem 9. *If the original system is zero-dimensional, the number of its solutions (counted with multiplicity and solutions at infinity) is precisely the cardinality of $\mathcal{A}$.*

Example 15: Consider the following system of equations [36]:

$$\begin{cases} cx + xy^2 + xz^2 - 1 = 0, \\ cy + yx^2 + yz^2 - 1 = 0, \\ cz + zx^2 + zy^2 - 1 = 0. \end{cases}$$

Then the leading monomials of the polynomials in the standard basis (using the `grevlex` ordering) are:

$$zy^3, \quad xy^2, \quad zx^2, \quad y^2z^2, \quad y^4, \quad z^5, \quad yz^4, \quad x^4, \quad xz^4, \quad yx^2, \quad xz^3y,$$

and the set of monomials $\mathcal{A}$ is:

$$1, \ x, \ x^2, \ x^3, \ y, \ xy, \ y^2, \ y^3, \ z, \ zx, \ zy, \ zyx, \ zy^2, \ z^2, \ z^2x, \ z^2y, \ z^2yx, \ z^3, \ z^3x, \ z^3y, \ z^4,$$

so the number of finite solutions of the system is 21 by Theorem 9.

Note that the leading monomials of the polynomials in the Gröbner bases using the `lex` ordering are

$$x, \quad y^2, \quad yz^7, \quad z^{14},$$

so in this case $\mathcal{A}$ is

$$1, \ z \ , \ \ldots \ , z^{13}, \ y, \ yz, \ \ldots \ , yz^6,$$

and its cardinality is also 21 as expected.

Experiences made with the linear algebra methods that we shall present in what follows show that for $c = \frac{2}{5}$, the number of real solutions is 7 out of the 21 above, while there is only one real solution when $c = 3$. □

The following sections will now consider some linear algebra techniques for getting information about the original polynomial system. We shall first present a way of counting the number of different solutions of the system and then how to approximate these solutions by computing the eigenvalues of certain matrices. We end up by providing a way of counting those roots of the system which are not solutions of a given polynomial.

4.3.1. Counting the number of different solutions. Once $\mathcal{A}$ is known, the following technique can be used to compute the number of different solutions of the initial system. First, compute the Jacobian determinant J of $f_1, \ldots , f_n$. Then, for each monomial m in $\mathcal{A}$, compute the normal form of mJ with respect to $(G,>)$ and store the coefficients in a column of a matrix $\mathcal{P}$ (in the same order as in $\mathcal{A}$).

Theorem 10. *The number of different solutions of the original system is the rank of $\mathcal{P}$.*

Example 16: Consider the following system:

$$\begin{cases} f := x_1x_2 - x_1^3 + x_2^2 - 2x_1^2x_2, \\ g := 2x_1^2 - x_2^2 + x_1x_2. \end{cases}$$

The Gröbner basis with respect to $>_{\texttt{grevlex}}$ is:

$$G = \{5x_2^4 - 32x_2^3 + 32x_1x_2 + 32x_2^2, 2x_1^2 - x_2^2 + x_1x_2, 4x_1x_2 + 4x_2^2 + x_1x_2^2 - 3x_2^3\}.$$

The leading terms are thus $5x_2^4, 2x_1^2$ and $x_1x_2^2$. And the set $\mathcal{A}$ is:

$$\mathcal{A} = \{1, x_2, x_2^2, x_2^3, x_1, x_1x_2\}.$$

Thus, the system has 6 solutions counted with multiplicities according to Theorem 9.

We compute the Jacobian determinant:

$$\begin{aligned} J &= \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{vmatrix} \\ &= -4x_2^2 - 8x_1x_2 + 4x_1^2x_2 + 5x_1^3 + 8x_1x_2^2 - 4x_1^2. \end{aligned}$$

Then, we compute the normal forms of mJ for all monomials m in $\mathcal{A}$:

$$15(2x_2^3 - 3x_1x_2 - 3x_2^2), \quad 12(x_2^3 - x_1x_2 - x_2^2), \quad \tfrac{144}{5}(x_2^3 - x_1x_2 - x_2^2), \ \tfrac{1728}{25}(x_2^3 - x_1x_2 - x_2^2), \quad 6(x_2^3 - x_1x_2 - x_2^2), \quad \tfrac{72}{5}(x_2^3 - x_1x_2 - x_2^2).$$

This gives birth to the following matrix:

$$\mathcal{P} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -45 & -12 & -\frac{144}{5} & -\frac{1728}{25} & -6 & -\frac{72}{5} \\ 30 & 12 & \frac{144}{5} & \frac{1728}{25} & 6 & \frac{72}{5} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -45 & -12 & -\frac{144}{5} & -\frac{1728}{25} & -6 & -\frac{72}{5} \end{pmatrix},$$

the rank of which is 2. The original system has thus 2 different solutions by Theorem 10. □

4.3.2. Approximating the solutions. Now, to find approximate values of the solutions, construct a set of matrices $\mathcal{P}_{x_i}$ as above, except that the Jacobian determinant is replaced by x_i. Then use some numerical algorithm to compute the eigenvalues of $\mathcal{P}_{x_i}$.

Theorem 11. *The solutions of the original system are of the form $(\lambda_1, \cdots, \lambda_n)$, where λ_i is an eigenvalue of $\mathcal{P}_{x_i}$.*

This means that once the eigenvalues of the above matrices have been found, they have to be combined to decide what are the true solution of the system.

Example 16 continued: One can now construct two matrices by computing the normal forms of each monomial in $\mathcal{A}$ multiplied by $x_i, i = 1, 2$. The normal forms with respect to x_1 are

$$x_1,\ x_1x_2,\ -4x_1x_2-4x_2^2+3x_2^3,\ \frac{16}{5}(x_2^3-x_1x_2-x_2^2),\\ \frac{1}{2}(x_2^2-x_1x_2),\ 2x_1x_2+2x_2^2-x_2^3,$$

and with respect to x_2:

$$x_2,\ x_2^2,\ x_2^3,\ \frac{32}{5}(x_2^3-x_1x_2-x_2^2),\ x_1x_2,\\ -4x_1x_2-4x_2^2+3x_2^3,$$

from which we deduce the following matrices:

$$\mathcal{P}_{x_1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & -\frac{16}{5} & \frac{1}{2} & 2 \\ 0 & 0 & 3 & \frac{16}{5} & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & -\frac{16}{5} & -\frac{1}{2} & 2 \end{pmatrix},$$

$$\mathcal{P}_{x_2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{32}{5} & 0 & -4 \\ 0 & 0 & 1 & \frac{32}{5} & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{32}{5} & 1 & -4 \end{pmatrix}.$$

The eigenvalues of $\mathcal{P}_{x_1}$ are 0 and $\frac{6}{5}$, and that of $\mathcal{P}_{x_2}$ are 0 and $\frac{12}{5}$. According to Theorem 11, the solutions of the system are to be found by combining these eigenvalues. We then readily check that $(0,0)$ and $(\frac{6}{5}, \frac{12}{5})$ are the two different solutions of the original system. □

4.3.3. Trace matrices. We now remove the hypothesis made above on the number of equations of the initial system. Suppose given $(G, >)$ and the normal form monomials $\mathcal{A} = (m_1, \ldots, m_D)$ as above. For any $m \in \mathcal{A}, m = x_1^{i_1} \cdots x_n^{i_n}$, define the matrix $\mathcal{P}_m$ by:

$$\mathcal{P}_m = \mathcal{P}_{x_1}^{i_1} \cdots \mathcal{P}_{x_n}^{i_n}.$$

Also, this notion extends to any polynomial h by summing the matrices obtained for each monomial of the normal form of h, forming matrix $\mathcal{P}_h$. Define then the trace $\mathrm{Tr}(h)$ of h to be the trace of $\mathcal{P}_h$.

Now construct matrices T_h as follows:

$$T_h = \begin{pmatrix} \mathrm{Tr}(h\, m_1\, m_1) & \cdots & \mathrm{Tr}(h\, m_1\, m_D) \\ \vdots & & \vdots \\ \mathrm{Tr}(h\, m_D\, m_1) & \cdots & \mathrm{Tr}(h\, m_D\, m_D) \end{pmatrix}.$$

Theorem 12. *The rank of the matrix T_h is the number of solutions of the initial system which are not roots of h. Thus the number of solutions of the initial system is the rank of T_1.*

An interesting feature is also that the kernel of T_1 is a polynomial system with the same solution set as the initial one, but with a Jacobian matrix having full rank on any solution.

Example 16 continued: To build the trace matrix T_1, we need for instance to compute $\mathrm{Tr}(m_2 m_6)$. Here, $m_2 = x_2$ and $m_6 = x_1x_2$, and thus

$$\mathrm{Tr}(m_2 m_6) = \mathrm{Tr}(\mathcal{P}_{x_1}\mathcal{P}_{x_2}^2) = \frac{864}{125}.$$

The trace matrix is then:

$$T_1 = \frac{6}{15625}\begin{pmatrix} 15625 & 6250 & 15000 & 36000 & 3125 & 7500 \\ 6250 & 15000 & 36000 & 86400 & 7500 & 18000 \\ 15000 & 36000 & 86400 & 207360 & 18000 & 43200 \\ 36000 & 86400 & 207360 & 497664 & 43200 & 103680 \\ 3125 & 7500 & 18000 & 43200 & 3750 & 9000 \\ 7500 & 18000 & 43200 & 103680 & 9000 & 21600 \end{pmatrix}.$$

We find that its rank is 2, so the system has 2 solutions by Theorem 12, and its kernel is

$$\left\{x_1\left(x_2 - \frac{12}{5}\right), x_2 - 2x_1, x_2^2 - \frac{24}{5}x_1, x_2^3 - \frac{288}{25}x_1\right\},$$

from which we retrieve the solutions. □

5. Symbolic system solving in the real case

Less results are known for dealing with the real solutions of polynomial systems. In particular, shooting directly for the real solutions (*i.e.* without computing the complex ones) is hard. We present here techniques for computing the number of real solutions of a system of algebraic equations.

A survey of computer algebra techniques in the real case can be found in [32]. New advanced results in the field can be found in the recent thesis [85].

In the following sections, we review how to count the number of real solutions in the univariate case (section 5.1). Another important issue is the way algebraic numbers are coded in computer algebra systems where exact representation is required. We examine two different types of coding in section 5.2, in conjunction with results known for bounding the size of real zeros of a univariate polynomial. Finally, section 5.3 gives results more results for counting the number of real solutions in the multivariate case in a linear algebra setting.

5.1. Univariate polynomials

Let $P = \sum_{i=0}^{d_P} a_i x^{d_P - i}$ and $Q = \sum_{i=0}^{d_Q} b_i x^{d_Q - i}$ be two univariate polynomials. Let us define a sequence of polynomials (the *Sturm sequence*) as follows: let P_{i+1} be the opposite of the remainder of the division of P_{i-1} by P_i, with $P_0 = P$ and $P_1 = QP'$. Let also $k = \sup(i \ / \ P_i \neq 0)$ $(k < d_P + d_Q)$. Let $a \in \mathbb{R}$ be such that $P(a) \neq 0$ and note $V_{\mathrm{Stu}(P,Q)}(a)$ the number of sign changes in the Sturm sequence $P_0(a), \ldots, P_k(a)$, not taking into account zero values.

Theorem 13. ([101]) *Let $V_{\mathbb{R}}(P)$ be the number of real zeros of P. Assume we have two reals $a < b$, then (# means cardinality)*

$$\begin{aligned} V_{Stu(P,Q)}(a) - V_{Stu(P,Q)}(b) = \\ \#\{x \in V_{\mathbb{R}}(P) \cap]a, b[\ / \ Q(x) > 0\} \\ - \#\{x \in V_{\mathbb{R}}(P) \cap]a, b[\ / \ Q(x) < 0\}. \end{aligned}$$

In particular,

$$\#\{x \in V_{\mathbb{R}}(P)\} = V_{Stu(P,1)}(-\infty) - V_{Stu(P,1)}(+\infty).$$

The main problem with this sequence is that even if the coefficients of P and Q are integers, the Sturm polynomials may have coefficients in $\mathbb{Q}$, so that the method may be inefficient in practice. A generalisation, known as the Sturm-Habicht sequence, has been constructed which enjoys the same properties as above. The readers are referred to [86] for the details on how to build this sequence. Sturm-Habicht polynomials share many common features with sub-determinants of the Sylvester matrix we have seen.

Sturm-Habicht sequences also allow to solve sign determination problems such as the following one. Let $P, Q_1, \ldots, Q_k$ be polynomials in $\mathbb{Z}[x]$. Determine which are the sign conditions verified by the polynomials $Q_1, \ldots, Q_k$ when evaluated on the real roots of P. See [32] for more.

Example 17: Here is an application of Theorem 13. Consider the following system:

$$\begin{cases} P := 48x^2 - 48x + 4y^2 + 9 = 0, \\ Q := x^2 + y^2 - 1 = 0. \end{cases}$$

Question is: do P and Q intersect? Consider P as a polynomial in y and compute the Sturm sequence of P and Q:

$$P_0 = P, \quad P_1 = 8y(x^2 + y^2 - 1), \quad P_2 = -P,$$
$$P_3 = y(88x^2 - 96x + 26), \quad P_4 = 48x^2 - 48x + 9.$$

Let s_1 be the sign of $y(88x^2 - 96x + 26)$ and s_2 the sign of $48x^2 - 48x + 9$. Then we have the following sign sequences:

	P_0	P_1	P_2	P_3	P_4
$y = -\infty$	$+$	$-$	$-$	$-s_1$	s_2
$y = \infty$	$+$	$+$	$-$	s_1	s_2

Thus, if s_1 and s_2 are negative, we have $V_{\mathrm{Stu}(P,Q)}(-\infty) - V_{\mathrm{Stu}(P,Q)}(\infty) = 2$. Since P has only two real roots, we deduce that the number of real roots of P at which Q is positive is 2. Thus if s_1 and s_2 are negative for certain values of x and y, P and Q will intersect. s_2 is negative for $x \in \left]\frac{1}{4}, \frac{3}{4}\right[$ so we look for roots of $R = 88x^2 - 96x + 26$ in this interval. The Sturm sequence of R and 1 is:

$$R_0 = R, \quad R_1 = 176x - 96, \quad R_2 = \frac{2}{11},$$

with the following sign sequences:

	R_0	R_1	R_2
$x = \frac{1}{4}$	+	−	+
$x = \frac{3}{4}$	+	+	+

Thus, $V_{\mathrm{Stu}(R,1)}(\frac{1}{4}) - V_{\mathrm{Stu}(R,1)}(\frac{3}{4}) = 2$ and the two roots α, β of R are in $\left]\frac{1}{4}, \frac{3}{4}\right[$. In addition, R is negative in $]\alpha, \beta[$ and we conclude that there are values of x, y for which both s_1 and s_2 are negative. Thus P and Q intersect. $\square$

5.2. Real numbers coding

Computer algebra requires exact representation of input data and of the possible solutions, *i.e.* infinite precision rational numbers (for linear systems) and suitable representations of real and complex algebraic numbers (non-linear systems). This is what we examine in this section.

5.2.1. Isolating intervals. The traditional way of coding real numbers is through the use of isolating intervals [62]. A point p of E^r is *algebraic* if each of its components is a real algebraic number. A real algebraic number γ is exactly represented by its minimal polynomial $M(x)$ and an isolating interval for a particular root of $M(x)$. A representation for an algebraic point p of E^r is: the minimal polynomial and isolating interval representation of one of its components, plus a r-tuple of representations of the coordinates of p. For instance, let $M(x)$ be the irreducible polynomial $x^3 + x + 1$, α the unique root of M that lies in the interval $(-1, 0)$ and $\beta = \frac{\alpha}{3}$. Then the algebraic point (α, β) is exactly represented by the triple:

$$\left[M(x), (-1,0), \left(x, \frac{x}{3}\right)\right].$$

5.2.2. Bounds on roots. Several results are known for bounding *a priori* the size of a real zero of a univariate polynomial, which may be used to obtain isolating intervals. The best known bounds are due to Cauchy. Given a univariate $f = \sum_{i=0}^{n} a_i X^i \in \mathbb{Q}[X]$ with $a_n \neq 0$, then the absolute value of every real zero of f is bounded by the following expressions:

$$C_1 = 1 + \max\left(\frac{|a_i|}{|a_n|}, 0 \le i \le n-1\right),$$

$$C_2 = \max\left(1, \sum_{i=0}^{n-1} \frac{|a_i|}{|a_n|}\right).$$

If a Sturm-Sylvester sequence has been computed for (f, f'), then one can evaluate the number of sign changes of this sequence at the endpoints obtained from the starting interval $[-C_i, C_i], i = 1, 2$ by successive bisection until one has found a finite sequence of pairwise disjoint open intervals with rational endpoints each of which contains exactly one real root of f. We thus can find isolating intervals for the real roots of f, which yield their coding. In addition, iterating the bisection of intervals, good numerical approximations of the roots can be obtained if needed.

Other results are known that can help finding isolating intervals. A theorem of Lagrange gives upper and lower limits on the positive roots of a polynomial, as a function of the coefficients of the polynomial. A method due to Newton gives the upper limit on the positive roots of a univariate polynomial f using its derivatives. Mignotte upper bounds the distance between roots as a function of the coefficients of f and of the discriminant of f and f'. Davenport expresses this distance inequality as a function of the largest module of a root of some polynomial derived from f. We won't go into more details here. The interested reader is referred to [60] for more on the criteria known to characterise the magnitude of roots and distance between roots of univariate and multivariate polynomial equations.

5.2.3. Coding à la *Thom.* Another possibility for representing real numbers is known as *Thom's coding* [21]. The idea is to use a lemma due to Thom: given a polynomial P of $\mathbb{R}[x]$, no two different real roots of P give the same signs to the derivatives of P. Thus the coding of a root α of

P (degree n) is given by $[P; \varepsilon_{n-1}, \ldots, \varepsilon_1]$, where $\varepsilon_i \in \{+, -, 0\}$ and the sign of $P^{(i)}(\alpha)$ is ε_i. This coding may obviously be obtained using Sturm-Habicht sequences.

Though Thom's coding is in practice less efficient than the use of isolating intervals, it does not need to be refined and also it provides the only way for working over more general base fields than the one considered here.

5.3. Trace matrices: number of real solutions

As in the complex case, linear algebra methods allow to give information on the real roots of a polynomial system. Assume that we are using the notations introduced in section 4.3 on trace matrices. Define the signature of a real symmetric matrix as the difference between the numbers of positive and negative squares once the associated quadratic form has been diagonalised. We then have the following result, known as Hermite's theorem:

Theorem 14. ([38, 68]) *The signature of T_h is equal to the number of real roots of $f_1, \ldots, f_n$ at which h is strictly positive minus the number of real roots of $f_1, \ldots, f_n$ at which h is strictly negative. In particular, the number of real roots of $f_1, \ldots, f_n$ is the signature of T_1.*

Example 16 continued: We now compute the eigenvalues of the trace matrix T_1. They are equal to

$$\frac{3}{15625}(640039 \pm \sqrt{377128359021}),$$

and thus evaluate roughly to 4.98 and 240.8. Thus the signature of T_1, the number of real roots of the system by Theorem 14, is 2 as expected.

Let us now fix a polynomial $h = x_1 - x_2 + 1$. Then we find that the rank of T_h is equal to 2, which indeed is the case since the solutions of the initial system are not zeros of h (Theorem 12). The eigenvalues of T_h are

$$\frac{6}{78125}(-280957 \pm \sqrt{119588788974}).$$

They evaluate to 4.98 and -48.14, and so the signature of T_h is 0, as expected since $h(0,0) = 1$ and $h(\frac{6}{5}, \frac{12}{5}) = -\frac{1}{5}$. $\square$

When the polynomial system has as many equations as unknowns, another method exists based on Bezoutians for computing the number of real solutions [32]. Roughly speaking, for $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, write

$$\Gamma_{i,j}(\mathbf{x}, \mathbf{y}) = \frac{f_i(y_1, .., y_{j-1}, x_j, .., x_n) - f_i(y_1, .., y_j, x_{j+1}, .., x_n)}{x_j - y_j}.$$

The *Bezoutian* of $f_1, \ldots, f_n$ is determinant of the matrix whose coefficients are the $\Gamma_{i,j}(\mathbf{x}, \mathbf{y})$. The idea then is to construct a matrix $\mathcal{B}$ based on the coefficients in the normal form of the Bezoutian, with respect to the Gröbner basis of $f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}), f_1(\mathbf{y}), \ldots, f_n(\mathbf{y})$. Then the number of real roots of the initial system is the signature of the matrix $\mathcal{P}_J \mathcal{B}$, where J is the Jacobian determinant. And the signature of $\mathcal{P}_{Jh} \mathcal{B}$ is the number of roots of the system at which h is positive minus the number of roots at which it is negative.

6. Applications

Applications are what motivated the search for more efficient algorithms dealing with algebraic sets. Several types of applications issued from computer vision, robotics and signal processing were used to evaluate the performances of polynomial system solvers.

A well-known example is that of Stewart platforms. The Stewart platform is a parallel manipulator with six prismatic joints connecting two rigid bodies. The base body is fixed, while the top one (the platform) is moving in 3-space, controlled by the lengths of the joints. The platform has one degree of freedom per joint; its position and orientation is specified by six parameters, three for the orientation and three for the position in space. In inverse kinematics, the joint parameters must be computed from the platform position and orientation. In forward kinematics, the displacement is to be found given the leg lengths.

The inverse kinematics problem has been studied for instance in [32]. In forward kinematics, it has been found that given a set of leg lengths, the number of stable positions (over the complex numbers) of the platform is 40. A first proof was found by Ronga and Vust [84] using intersection theory. A simpler proof was then given by [65] using exterior calculus. The problem was again revisited in [26] using sparse resultants and by [85] recently using new algorithms in the real case. This last author produces an example of a situation where the platform can have as many as 24 different real positions.

Again in [85], the Birkhoff interpolation problem is shown to be solvable in situations that had previously defied attack. It consists in deciding if the knowledge of a finite number of values of a function $f : \mathbb{R} \to \mathbb{R}$ or of some of its derivatives is sufficient to determine a polynomial p interpolating f. The problem is shown to be reducible to the problem of determining whether some hypersurface admits or not real points with all coordinates non-zero.

Many other problems have been approached using computer algebra techniques, like motion from point matches [26] or the determination of low-pass filters satisfying a given set of conditions [85]. Also, in the context of the description of two-dimensional grey-valued images, the application of invariance theory often involves algebraic sets [66, 88].

In the following sections, we briefly present several other interesting applications. In particular, we look at the construction of aspect graphs in section 6.1, at the piano mover's problem (section 6.2), at the implicitisation of parametric surfaces (section 6.3) and at the determination of 3D information from 2D images (sections 6.4 and 6.5).

6.1. Computation of aspect graphs

Informally, the *aspect graph* [52] is a qualitative viewer-centered representation that enumerates all the topologically distinct views (or aspects) of an object. More formally, choosing a camera model and a viewpoint determines the aspect of an object (*i.e.* the structure of the observed line-drawing). The range of possible viewpoints can be partitioned into maximal connected sets (regions) that yield identical aspects. The change in aspect at the boundary between regions is called a *visual event*. The maximal regions and the associated aspects form the nodes of an aspect graph, whose arcs correspond to the visual event boundaries between adjacent regions.

After presenting some of the properties of aspect graphs of algebraic objects, we shall describe an algorithm for constructing aspect graphs, due to Rieger [76], which is based on some of the symbolic techniques we have reviewed in this paper.

6.1.1. Introduction. We use the hypothesis that the building blocks of the objects in the scene are semi-algebraic bounded closed subsets of smooth algebraic surfaces. We also suppose that these building blocks are such that:

- they either intersect transversally or don't intersect;
- no four surfaces intersect at the same point.

This situation is often referred to as *generic*. So we place ourself in this situation. Note that the assumption that the surfaces be smooth can somewhat be relaxed: they can be piecewise-smooth, with transversal self-intersection (double curve) and isolated pinchpoints (points of the double curve at which the two tangent planes coincide) and triple points (this is the limit case: beyond this limit, the notion of aspect graph has no meaning). But the hypotheses we use here are very sufficient for our purpose.

With these hypotheses, one can show that the different contours and the visual event surfaces partitioning viewpoint space are semi-algebraic and each step of the construction of the aspect graph can be realized in an exact way using effective methods of real algebraic geometry [108]. This exact construction of aspect graphs has been treated by different authors: [87] for solids of revolution, [74] for smooth algebraic surfaces and [76] for piecewise-smooth algebraic surfaces. We give here the flavour of these methods.

For sake of simplicity, we shall consider a unique smooth surface M of defining equation $f(x, y, z) = 0$. Let $\mathcal{V}$ be the viewspace considered and $\mathcal{B}$ the bifurcation set, *i.e.* the set of viewpoints for which the projection of M is degenerate. Let $\mathbf{w}$ be a direction of projection, $t \mapsto \mathbf{l}(t)$ a viewing ray and $K(t) = f \circ \mathbf{l}(t)$, where t is a parameter along the ray. If perspective projection is assumed then $\mathbf{l}(t) = \mathbf{p} + (\mathbf{w} - \mathbf{p})t$.

The viewpoints observing degenerate views correspond to certain rays having particular geometric contacts with the surface. This gives birth to a set of *recognition equations* $\tilde{\mathcal{B}}_j \subset \mathcal{V} \times \mathbb{R}^{m_j}$, where $\mathbb{R}^{m_j}$ is the space of surface parameters [70]. For instance, a whole set of such viewpoints (a visual event surface) occurs when the viewline has contact of order 4 with M (the so-called *flecnodal* event). The corresponding recognition equation is:

$$\tilde{\mathcal{B}}_1 = \{(\mathbf{p}, \lambda) \in (\mathcal{V} \times \mathbb{R} \;/\; K(\lambda) = 0, \\ K'(\lambda) = 0, K''(\lambda) = 0, K'''(\lambda) = 0\}.$$

By projecting the $\tilde{\mathcal{B}}_j$ onto $\mathcal{V}$, we obtain a set of visual event surfaces $\mathcal{B}_j$ the union of which is $\mathcal{B}$.

6.1.2. Rieger's method. The exact construction of the aspect graph goes as follows. Some of the techniques used come from complex algebra (*e.g.* elimination) and others from real algebra (*e.g.* real root isolation). Note that working with real points complicate matters a lot: certain techniques used for determining properties of a real algebraic set require that the complex zero-set of the defining equations of this real set does not have excess components or components at infinity.

Here we describe one of the algorithms given in [76]. Let $\hat{\mathcal{B}}_j$ be the real algebraic sets obtained by eliminating the surface parameters among the defining equations of $\tilde{\mathcal{B}}_j$. Let also $\hat{\mathcal{B}} = \cup\hat{\mathcal{B}}_j$. The algorithm then goes as follows:

- *Obtain defining equations for the sets $\hat{\mathcal{B}}_j$.* If the recognition equations have been chosen appropriately, then a standard elimination technique (resultants - section 3 - or Gröbner bases - section 4) can be used.
- *Determine the connected components of $\mathcal{V} - \hat{\mathcal{B}}$ and their adjacencies.* This step is based on a modified version of Collins' cylindrical algebraic decomposition algorithm [20].
- *Remove the branches of $\hat{\mathcal{B}} - \mathcal{B}$ and merge the regions separated by these branches.* The components of $\hat{\mathcal{B}}_j - \mathcal{B}_j$ are the projections of complex solutions of the recognition equations and have to be removed to recover $\mathcal{B}_j$. A key property here is that the boundary of $\mathcal{B}_j$ in $\hat{\mathcal{B}}_j$ is contained in some other component $\mathcal{B}'_j$. This step relies upon Hermite's theorem (section 5.3) and isolating intervals (section 5.2).

This procedure has recently been updated [75] to account for the singly exponential stratification algorithm of [19, 90].

6.2. The piano mover's problem

Another application that makes use of the methods described above is known as the "piano mover's problem". One formulation of this problem is as follows. Let K be a compact semi-algebraic set (the piano) of $\mathbb{R}^3$ and U be an open semi-algebraic set of $\mathbb{R}^3$. The goal then is to:

- Decide whether there exists a continuous motion moving K from an initial position K_0 to a final position K_1 in U;
- If the above is true, then compute such a motion.

The complete treatment of this problem in the context of real algebraic geometry is presented in [10].

A position of K may be described by a translation followed by a rotation (starting from K_0), *i.e.* by a point of $S = \mathbb{R}^3 \times \mathbb{P}^3(\mathbb{R})$. S is a 6-dimensional algebraic variety and the set S' of allowed positions for K defined by:

$$S' = \{s \in S \;/\; s(K) \subset U\}$$

is an open semi-algebraic set of S. Consider the diagram (where π and Φ are respectively the projections onto the first and second factors):

$$\begin{array}{ccc} S \times \mathbb{R}^3 & \xrightarrow[\Phi]{} & \mathbb{R}^3 \\ {\scriptstyle\pi}\downarrow & & \\ S & & \end{array}$$

Then the set $A = \Phi^{-1}(\complement U) \cap (S \times K)$ is a semi-algebraic set of $S \times \mathbb{R}^3$ and its image by π is $S \setminus S'$. The piano mover's problem may then be reformulated as follows. Given a semi-algebraic set A of $\mathbb{R}^3 \times \mathbb{P}^3(\mathbb{R}) \times \mathbb{R}^3$, describe the connected components of $B = \mathbb{P}^3(\mathbb{R}) \times \mathbb{R}^3 \setminus \pi(A)$ and find explicitly a path between any couple of points in the same component of B. Such a description can be achieved using a cylindrical algebraic decomposition of B (Collins' algorithm [20]).

6.3. *Implicitisation*

Two widely used forms of algebraic surfaces are the parametric and the implicit forms. Surfaces that can be parameterised by rational algebraic functions are called *rational*. It is well-known that every rational surface admits an implicit form over the complexes, while the converse is not true. Both representations have advantages and disadvantages for solid modelling and computer vision. For instance, parametric equations facilitate point generation and have an order of flexibility [41]. But such operations as point appartenance, fitting, and representation of space curves are more easily done with implicit equations [115]. Some operations can also be greatly simplified when both representations are available, like the intersection of two surface patches or the triangulation for curved object display. This is why conversion algorithms between these two representations have been studied extensively over the past few years [59, 109]. We concentrate here on computing the implicit form of a surface given in parametric equations, a process known as *implicitisation*.

A first used technique is based on elimination theory (see [41]). Consider the following parameterisation of a surface S given in homogeneous form:

$$F(u,v) = (X(u,v), Y(u,v), Z(u,v), T(u,v)).$$

The rough idea is to set up equations of the form:

$$\begin{cases} yX(u,v) - xY(u,v) = 0, \\ zX(u,v) - xZ(u,v) = 0, \\ tX(u,v) - xT(u,v) = 0, \end{cases} \qquad (3)$$

where x, y, z, t are scalar variables, and to eliminate u, v from the above equations to obtain a resultant implicit function f of x, y, z, t. Such methods either produce a function f that is not the implicit equation of S (it can contain extraneous factors which may be very time consuming to separate) or are limited to special parameterisations (like tensor product surfaces). A second technique was based on the use of Gröbner bases and computes a canonical representation of the ideal generated by the parametric equations [15]. It may be an extremely slow process in practice.

The above two methods fail altogether in the presence of base points, *i.e.* points (s_0, t_0) such that:

$$X(s_0,t_0) = Y(s_0,t_0) = Z(s_0,t_0) = T(s_0,t_0) = 0.$$

These points occur frequently in practice and they blow up to rational curves on the surface (*seam curves*), so that it is quite important to deal efficiently with them. Some devices have been developed to reparameterise surfaces so as to eliminate some of their base points, but are currently limited to low-degree parameterisations. More recent techniques, like [59], deal directly with base points by introducing an efficient perturbation of one of the equations of System (3). The implicit equation of S is then shown to be contained in the lowest degree term of the resultant f (in terms of the perturbing variable). However, this term also contains an extraneous factor which turns out to be the projection of the seam curves and can be used to compute a rational parameterisation of these curves. Recent works have tried to lower the size of the matrix from which the resultant is formed [89].

6.4. *View consistency*

Constraints derived from assumptions about models and projections can be translated into an algebraic form. These algebraic constraints are typically nonlinear equations and inequalities (we assume that we deal with ideal images in which image coordinates are exactly known). The consistency of two views can be established by deciding whether algebraic constraints corresponding to the two views are consistent [4]. A key result is that relatively few groups of constraints are sufficient to prove the inconsistency between two views. If two views are consistent, it is possible to determine the transformation between views and extend the model to include explicit 3D constraints. Imprecise data and tolerances can also be dealt with.

A Gröbner basis algorithm is used to check whether the algebraic equations are consistent or not [14]. In case the system is consistent, their basis embodies all the information about the model which can be extracted from images. This information may be stored for subsequent manipulation when equations corresponding to additional constraints are introduced [4].

6.5. 3D from 2D

Recovering and manipulating 3D information from 2D projections has been extensively studied in the vision literature [45]. Many questions involved are variants of the following: given N views and n 3D objects matched across the views, what exactly can be recovered of the 3D structure and what are the different ways to achieve that. Related to this is the work of [100] using the correspondences of classical algebraic geometry and the dimensions of flag spaces.

Partial answers have been given to this question in special cases, but a general framework has only been presented recently [112]. The high-level idea is to set up the equations describing the projections of the 3D objects onto corresponding 2D points and then to view the problem as an elimination problem. Depending on what is to be searched, the variables of the above equations are grouped into three categories: those to be eliminated, those to treat as constants, and those that will be the variables of the implicit form of the original equations. Then the elimination process is performed using Gröbner bases. Let us give the conditions in terms of the number of views, of image points and of camera parameters for this method to succeed.

Suppose for instance that a pinhole camera model is used and that n 3D points are given (similar arguments can be developed for lines - see [112]). Five points (in general position) are needed to form a projective basis of 3D space, and thus there are $3n-15$ space variables. Since the 3 by 4 projection matrix is defined up to scale, there are $11N$ camera parameters. Finally, there are $2nN$ image variables.

If the goal is to reconstruct the 3D points, then the camera parameters have to be eliminated along with the space parameters, except one (say the z coordinate). This is only possible if

$$2nN > 11N + 3n - 16.$$

For instance, if $N=2$, the minimal number is $n=7$ and the degree of the implicit equation in z obtained after elimination gives an upper bound on the number of solutions to the reconstruction problem (3 in this case). If $n=6$ ($N=2$), then there are 24 equations and 24 variables to eliminate, so elimination cannot be carried out.

If now we are interested in matching constraints (functions of image coordinates that vanish for all tuples of corresponding points), then the $11N$ camera parameters have to be treated as constants. For this, we need only consider the case $n=1$. The number of equations is $2N$ and the 3 space coordinates have to be eliminated. This is only possible if

$$2N > 3.$$

Thus, when $N=2$, there is only one constraint (the epipolar constraint). When $N=3$, there are 3 independent constraints. The computation of the Gröbner basis in this case also shows that 3 views admit sets of four trilinear invariant functions (27 coefficients overall) that can be solved linearly from 7 corresponding points across the views.

Finally, if one wants to know what can be obtained from a single view, the space coordinates are treated as constants. Elimination is then done on the 11 camera parameters, and the condition to satisfy is:

$$2n > 11.$$

Thus, for $n \geq 6$, there are $2n-11$ invariant functions of image coordinates whose coefficients are polynomials of the space coordinates (several point configuration in space can share the same invariant function).

7. A word on complexity

Now, let us briefly discuss some aspects of the complexity of polynomial system solving, with exact and approximate computations. As we shall see in section 7.1, symbolic computation is in the worst case an exponential time procedure. The interested reader is invited to consult [6] for a survey of the different complexity bounds involved in exact computations. In section 7.2 we then present recent results showing that approximating the roots of a polynomial system can be done in polynomial time to an arbitrary precision.

7.1. Exact computations

The complexity of a Gröbner basis computation for a given system may vary enormously according to the term order chosen. Some heuristics are known but no definite theoretical results for the choice of a good term order for a given problem. Experience shows however that $>_{\mathtt{grevlex}}$ is generally the most efficient term ordering for Buchberger's algorithm. If numerical computation of individual solutions is needed, the reduced Gröbner basis (the unique Gröbner basis G such that the normal form of each $g \in G$ with respect to $G \setminus \{g\}$ is g and such that the coefficient of the leading term of g is 1) needs to be in triangular form, which is only guaranteed for the lexicographical term orders. Also, a method is known [27] for zero-dimensional systems to convert Gröbner bases with respect to any ordering by linear algebra methods into Gröbner bases with respect to the pure lexicographic ordering.

The degree of the intermediate polynomials may be doubly exponential in the number of variables and the worst-case complexity is d^{2^n}, where d is the maximum total polynomial degree and n is the number of variables. For zero-dimensional varieties a tight upper bound on the complexity is $d^{O(n)}$. Wu's method has been shown to be exponential in n and polynomial in the degrees. The parallel time complexity is polynomial [6].

7.2. Approximate root finding

Recently Steve Smale and Mike Shub [93] have developed a theory of computation for the task of approximately finding the roots of a polynomial. The main result is a precise bound on the number of iterations needed to find a root.

Let $d = (d_1, \ldots, d_n)$, where each d_i is a positive integer. Let $\mathcal{H}_{(d)}$ be the linear space of all maps $f : \mathbb{C}^{n+1} \to \mathbb{C}^n$, $f = (f_1, \ldots, f_n)$, where each f_i is a homogeneous polynomial of degree d_i. Let also $\mathbb{P}(\mathcal{H}_{(d)})$ be the associated projective space. Then the average number k' of arithmetic operations to find l approximate zeros ($1 \le l \le \prod_{i=1}^n d_i$) of $f \in \mathbb{P}(\mathcal{H}_{(d)})$ is [91]

$$k' \le c_1 l^2 N^4,$$

unless $n \le 4$ or some $d_i = 1$ in which case $c_1 l^2 N^5$ operations are needed, where N is the dimension of $\mathcal{H}_{(d)}$ as a complex vector space and c_1 is a universal constant (as well as all of the c_i in the rest of this section).

Several things need to be clarified. First, what exactly is meant by an approximate zero of f. A $z \in \mathbb{P}(\mathbb{C}^{n+1})$ is an approximate zero of f if Newton's method converges quadratically, immediately, to an actual zero ξ of f, starting from z. Given an approximate zero of f, an approximation to ε of an actual zero can be obtained with a further $\log|\log\varepsilon|$ number of steps. Second, the word average refers to some probability measure on $\mathbb{P}(\mathcal{H}_{(d)})$ that was developed in [91]. Third, N may usually be bounded by some function of n. For instance, if $d = (2, \ldots, 2)$, then one can see that $N \le n^3$.

The algorithm behind this result is a homotopy method with steps based on a projective version of Newton's method. Let f_0 and f_1 be two polynomial maps of $\mathcal{H}_{(d)}$ in n variables. Let $f_t = tf_1 + (1-t)f_0$. Let S_t be the set $f_t^{-1}(0)$. The interval $[0,1]$ is divided, $0 = t_1 < t_2, \ldots, < t_m = 1$, and the roots $S_{t_{i+1}}$ are found iteratively from the roots S_{t_i}, $1 \le i < m$. Let Δ_n be the subset of $\mathcal{H}_n$ of singular polynomial maps. Let ρ be the distance[4] from the arc $t \mapsto f_t$, $0 \le t \le 1$, to Δ_n. Given a starting point, it can be proved that the probability of failure σ (function of ρ and of the starting point) is such that [93]

$$\sigma \le c_2 \rho^2 N^3$$

in the general case, and the number of steps needed for the homotopy method to find an approximate zero to f is

$$k \le \frac{c_3 N^3}{\sigma}.$$

Note that the number of arithmetic operations of each projective Newton step can be bounded by $c_4 N$, so that k and k' above are related by $k' \le c_4 N k$.

In [94], the result is extended: the number of projective Newton steps sufficient to find all the approximate zeros of f with probability x of success is

$$\frac{c_5 (\Pi_{i=1}^n d_i) \max(d_i) n^2 (n+1)(N-1)(N-2)}{1-x}.$$

8. Computational aspects

Before concluding, let us give a few pointers to the computational aspects of algebraic geometry. Several books do a great job of explaining in full length the many algorithms and methods touched upon in the following [2, 22, 62]. Note also that the WWW servers of the *Symbolic Mathematical Computation Information Center* and of the *Computer Algebra Information Network* contain a great deal of information pertaining to the algorithms we have discussed. We have made available the URLs of these servers on our WWW address[5], as well as the addresses where information can be obtained for the softwares described below.

8.1. Computer algebra systems

Recent years have seen the appearance of a large number of commercial softwares for doing computer algebra and manipulating algebraic equations. Among the general-purpose computer algebra systems (CAS for short), Reduce and Macsyma (advanced symbolic/numeric mathematical software) may not be called "classical" since their first versions appeared around 1968. More modern (starting 1980) are Maple, Mathematica, Derive, Magma (which does algebra, number theory and geometry), and Axiom (the heir of the Scratchpad system). A more dedicated platform is Ganith, toolkit developed by the Shastra group at Purdue University aimed at computing with and visualising algebraic equations. Note that Appendix C of [22] gives an introduction to the polynomial system functions of Maple, Mathematica and Reduce.

Among the public domain softwares, Fermat (system for Macs that excels at polynomial computations over the rationals, the integers and finite fields) and MuPAD (designed as a parallel CAS) are rather general purpose. There are also some really strong dedicated platforms. One of them is Macaulay [7], a CAS whose main task is the efficient calculation of Gröbner bases. It has solved many problems and provided many examples to the algebraic community. The Singular system is particularly suited for calculations in singularity theory and algebraic geometry. It can compute with ideals and modules generated by polynomials over polynomial rings. PoSSo is an experimental platform for solving polynomial systems. Another platform for the resolution of systems of algebraic equations, with a syntax close to that of Axiom is GB. The RealSolving toolbox [85] is devoted to the study of the real roots of zero-dimensional polynomial systems. It may be interfaced with GB for the calculation of Gröbner bases. Finally, Pari-GP, though mainly for number theory, has mathematical functions for handling polynomials.

The commercial systems have also been enhanced with a number of packages for specific tasks. For instance, `CoCoA` and `Cali` are two Reduce packages aimed at commutative algebra. Also, `Groebner` and `Wu` are two packages for the calculation of Gröbner bases and for the implementation of Wu's method. In the Maple world, the `IF` Maple package [23] is aimed at handling systems of polynomial inequations (methods like generalised Sturm sequences have been implemented). Also, `Casa` is a package for doing constructive algebraic geometry and `Grassmann` another Maple package to work with anti-commutative and non-commutative variables and functions.

8.2. Polynomial system solving capabilities

Of potential interest to the user are the different reports on the capabilities of computer algebra systems that have started to pop up in the literature. To the best of our knowledge, only one has been entirely devoted to this day to the study of the polynomial system solvers of general purpose CAS [34]. Another paper [36] has a section on their Gröbner bases implementation.

The strategy generally used by CAS for solving polynomial systems is to find a decomposition of the variety of solutions into irreducible components. Usually, a first try is made at solving the problem over the ground field (where the transformations do not involve introducing new algebraic quantities) and then over some extension of the ground field. This second step is usually encapsulated in the functional symbol $\mathrm{RootOf}(p(x), x)$ (as is the case in Maple), representing the sequence of solutions of the equation $p(x) = 0$. In [34], the author expresses the belief that dedicated platforms are presently not better than general-purpose CAS for solving polynomial systems because they lack of good factorisation algorithms.

Of the CAS considered (Axiom, Macsyma, Maple, Mathematica, MuPAD and Reduce), conclusion is made that Reduce and Maple offer satisfactory solve functionalities for advanced systems. Axiom has troubles with systems having infinitely many solutions and Macsyma and Mathematica do not handle well higher dimensional systems. The MuPAD system solver is in rudimentary state. In terms of Gröbner bases capabilities, the standard packages of Mathematica and Maple are not very advanced but of comparable strength. Of course, public domain improved Gröbner implementations are available [37, 29]. Macaulay contains a fast Gröbner basis implementation but can only be used on homogeneous polynomials. The `groebner` Reduce package goes further than the implementations in other CAS. For instance, in the zero-dimensional case, conversion of bases is possible from any ordering to the pure lexicographical case.

9. Final comments

We have presented the different techniques for dealing exactly with polynomial systems: counting the number of real and complex roots, coding real roots, solving polynomial systems. We have given examples of the use of most of the methods introduced and then presented applications arising in computer vision and robotics. We have then dealt with some of the computational aspects of polynomial system solving, first by giving some bounds on the complexity of this task and next by presenting most of the available packages for doing symbolic computations on polynomials.

Our overview has obviously left aside many interesting topics. We now turn our attention to some of the topics that will undoubtedly witness many developments in the near future:

- **Combining numerical and symbolic techniques**. At several places, we have made clear how symbolic and numerical techniques could interact. For instance, turning a Gröbner basis computation into a linear algebra problem allows for stable numerical computations (eigenvalues, kernel, ...) on matrices. Also, the computational complexity of computing solutions of polynomial system entirely exactly may be prohibitive and we have to be satisfied with numerical values of the solutions that are accurate up to an arbitrary precision.

  The main problem arising in symbolic-algebraic computation is the size of the expressions arising during computations. If a symbolic computation concerning polynomial systems fails, this is more often due to storage limitations rather than time limitations. A combination of numerical and symbolic methods may in the near future yield a significant computational progress, as many people have recognised and advocated recently [12, 67].

- **The number of real zeros.** Describing the common zeros of a set of polynomials is known to be more difficult over non-algebraically closed fields. Global results (*i.e.*, results valid for a whole class of polynomial systems and not just for isolated examples) on the number of real zeros of a system of polynomials are recent and diverging, which shows that it may well be very difficult to give general statements.

  Up to recently, only isolated geometric polynomial problems were known to be *fully real*, in the sense that all of their solutions can be real. For instance, [83] investigated the number of conics tangent to five general conics in the real case. They proved that in fact all (*i.e.* 3264 as was found by de Jonquières in 1859, and again by Chasles in 1864 - see [116]) can be real. In a recent series of papers, F. Sottile [95, 96, 97] was the first to give a general statement in this direction: he produces large classes of fully real non-trivial enumerative problems. For instance, he proves in [96] that for any problem of enumerating lines in $\mathbb{P}^n$ incident on real linear subspaces in general position, all solutions can be real. He later expended this result to other classes of enumerative problems by giving a procedure to create new fully real problems from existing ones [97].

  Despite these "positive results", some negative ones have also popped up. Sottile [95] notes interestingly that these negative examples all seem to involve intersections of non general subvarieties (*i.e.*, they do not intersect in the way two random varieties would intersect). For instance, F. Klein [51] showed that at most $n(n-2)$ of the $3n(n-2)$ flexes on a real plane curve of degree n can be real. This result was recently given a modern and precise proof by [82], using results on singularities of maps. But these flexes are the intersection of the original curve with its Hessian determinant, which is not a general curve of degree $3(n-2)$. Also Khovanskii [50] showed that for systems of polynomials with few monomials on a complex torus, the real zeros represent at most a small fraction of the complex zeros. However, these are not generic hypersurfaces with given Newton polytope.

Finally, some results are known on the expected or average number of real roots of a random polynomial systems. First, Kac proved in 1943 that the expected number of real roots of a degree d univariate polynomial is asymptotic to $\frac{2}{\pi}\log d$ using an appropriate probability measure [48]. Also, it is known that given a real $f \in \mathbb{P}(\mathcal{H}_{(d)})$, the average number of real zeros of the system is $\sqrt{\mathcal{D}}$, where $\mathcal{D} = \Pi_{i=1}^{n} d_i$, using the notations and probability measure of section 7.2. This result was proved in [53] in the case that all of the d_i's are the same and in [92] in the full case.

- **Multiplicity-free deformations.** In the context of real intersection theory, the ideas developed by Sottile consist in deforming general intersection cycles (formal sums of irreducible varieties) into unions of simpler cycles. This technique of multiplicity-free deformations may have applications beyond the existence of real solutions. When the deformations are explicitly described (which is the case in most known results), it may be possible to obtain explicit solutions to the enumerative problem using continuation methods of numerical analysis to follow real points in the degenerate configuration backwards along the deformation. Algorithms to accomplish this have recently been developed in the case of intersecting hypersurfaces in a complex torus [42]. Bridging the gap between symbolic computation and intersection theory will undoubtedly be a main challenge in the future.

To conclude, let us briefly elaborate on the importance of these possible future developments for such fields as computer vision or robotics. Since most of the objects used in these fields are either algebraic or well-approximated by algebraic patches, many vision and robotics problems boil down to solving a polynomial system or describing the components of some algebraic objects. More often than not, these polynomial systems come from non-generic geometric situations, meaning that some of the objects involved do not intersect transversally.

If the observation made by Sottile [95] (that in such situations the number of real zeros is a small fraction of the total number of roots) turns out to be true in general then it is of the utmost importance to be able to directly shoot for the real solutions. The author has witnessed this phenomenon in connection with the construction of aspect graphs [69], where polynomial systems arise from imposing high orders of contact between a line or a plane and an algebraic surface. These situations are in no way generic and indeed most of the construction time was spent computing complex roots that we did not want, since only a few per cent turned out to be real. The experiments we made showed that the number of different - real - aspects of an algebraic surface of degree $d = 4$ was of the order of a few dozens, while the best upper bound on the number of aspects that we could compute for this value of d was of the order of fifty millions (as a result of combining many "non-generic" polynomial systems).

It is thus clear that avoiding complex roots would really save a lot of computation time. Also, knowing in advance what is the maximum amount of real zeros that one can expect would provide first a measure of the complexity of constructing the object representation and second a way of asserting the potential of this representation in computer vision (for indexing purposes one can manage a few dozens of aspects per object but not a few millions). Of course, another problem raised by the matching / indexing issue is how to compare and relate "ideal" information with information extracted from images. For this, more work has to be done to precisely know how an algebraic object evolves when the coefficients of its defining equations are perturbed. In particular, one could benefit from an understanding of how the shape evolves (with an adequate definition of shape).

Acknowledgements

The author wishes to thank one of the reviewers for the many detailed comments which helped improve the clarity and presentation of the paper.

Notes

1. The ideal generated by a set of polynomials $f_1, \cdots, f_r \in R$, R a commutative ring, is the smallest ideal containing the set $\{f_1, \cdots, f_r\}$. In other words, it is the set $\{\sum_{i=1}^{r} g_i f_i \ / \ g_i \in R\}$.
2. Note that in Figures 1.a and 2, the coefficient of a monomial $x^i y^j$ is used to label the point of coordinates (i, j).
3. `http://www.loria.fr/~petitjea/linalg.html`.
4. This projective space distance is a fonction of the Riemannian (or Fubini-Study) distance. It would be cumbersome to define this distance in more details here so the reader is kindly referred to [92].
5. `http://www.loria.fr/~petitjea/algebra.html`

References

1. W. Adams and P. Lousraunau. *An Introduction to Gröbner bases*, volume 3 of *Graduate Studies in Mathematics*. Oxford University Press, 1994.
2. D. Arnon. *Computational Methods in Real Algebraic Geometry*. Academic Press, 1989.
3. D. Arnon. Geometric reasoning with logic and algebra. In Deepak Kapur and Joseph Mundy, editors, *Geometric Reasoning*, pages 37–60. MIT Press, 1989. Proceedings of the International Workshop on Geometric Reasoning.
4. M. Barry, D. Cyrluk, D. Kapur, J. Mundy, and V.D. Nguyen. A multi-level geometric reasoning system for vision. *Artificial Intelligence*, 37:291–332, 1988.
5. J. Barwise. An introduction to first-order logic. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 5–46. North Holland, 1977.
6. D. Bayer and D. Mumford. What can be computed in algebraic geometry? In *Computational algebraic geometry and commutative algebra (Cortona, 1991)*, Sympos. Math., XXXIV, pages 1–48. Cambridge University Press, 1993.
7. D. Bayer and M. Stillman. Macaulay: a system for computation in algebraic geometry and commutative algebra, 1992. Computer software available via anonymous ftp from zariski.harvard.edu.
8. M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 32(2):251–264, 1986.
9. R. Benedetti, F. Loeser, and J.-J. Risler. Two bounds for the number of connected components of a real algebraic set. In *Real Analytic and Algebraic Geometry*, volume 1420 of *Lecture Notes in Mathematics*, pages 22–35. Springer-Verlag, 1990.
10. R. Benedetti and J.-J. Risler. *Real Algebraic and Semi-Algebraic Sets*. Hermann, 1990.
11. D.N. Bernstein. The number of roots of a system of equations. *Functional Analysis and Applications*, 9(2):183–185, 1975.
12. D. Bini and V. Pan. *Numerical and Algebraic Computations with Matrices and Polynomials*. Birkhäuser, Boston, 1992.
13. J. Bochnak, M. Coste, and M.-F. Roy. *Géométrie algébrique réelle*, volume 12 of *Ergebnisse der Mathematik*. Springer-Verlag, 1987.
14. B. Buchberger. Gröbner bases: an algorithmic method in polynomial ideal theory. In N.K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. Reidel, Dordrecht-Boston-Lancaster, 1985.
15. B. Buchberger. Applications of Gröbner bases in non-linear computational geometry. In D. Kapur and J. Mundy, editors, *Geometric Reasoning*, pages 415–447. The MIT Press, 1989.
16. B. Buchberger, G.E. Collins, and R. Loos. *Computer Algebra - Symbolic and Algebraic Computation*. Springer-Verlag, 1983.
17. J. Canny and I. Emiris. An efficient algorithm for the sparse mixed resultant. In *Proceedings of* AAECC *(International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes), Puerto Rico*, volume 673 of *Lecture Notes in Computer Science*, pages 89–104. Springer-Verlag, 1993.
18. J.F. Canny. *The Complexity of Robot Motion Planning*. The MIT Press, 1988.
19. B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoretical Computer Science*, 84:77–105, 1991.
20. G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the Second GI Conference on Automata and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–163. Springer-Verlag, 1975.
21. M. Coste and M.-F. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *Journal of Symbolic Computation*, 5:121–129, 1988.
22. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics. Springer-Verlag, 1992.
23. F. Cucker and M.-F. Roy. IF, a package of Maple programs for computing with real algebraic numbers and working with real solutions of equations and inequalities, 1989. Version 0.1.
24. A.L. Dixon. The eliminant of three quantics in two independent variables. *Proceedings of the London Mathematical Society*, 6:49–69, 1908.

25. I. Emiris and J. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *Journal of Symbolic Computation*, 20(2):117–149, 1995.
26. I.Z. Emiris. *Sparse Elimination and Applications in Kinematics*. PhD thesis, University of California at Berkeley, 1994.
27. J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
28. W. Fulton. *Intersection Theory*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer-Verlag, 1984.
29. K. Gaterman. The `moregroebner` package, 1996. `http://www.zib.de/~bzfgatte/moregroebner.html`.
30. I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants and Resultants*. Birkhäuser, 1994.
31. L. González-Vega. Some examples of problem solving by using the symbolic viewpoint when dealing with polynomial systems of equations. In J. Fleischer, J. Grabmeier, F. W. Hehl, and W. Küchlin, editors, *Computer Algebra in Science and Engineering*, pages 102–116. World Scientific Publishing, 1995.
32. L. González-Vega. Symbolic recipes for polynomial system solving: real solutions. Lecture notes of the tutorial presented at ISSAC (International Symposium on Symbolic and Algebraic Computation), 1996. Zurich, Switzerland.
33. P. Gordan. *Vorlesungen über Invariantentheorie*. Verlag von Teubner, Leipzig, 1885–1887.
34. H.-G. Gräbe. About the polynomial system solve facility of Axiom, Macsyma, Maple, Mathematica, MuPAD and Reduce. Preprint, Universität Leipzig, August 1996.
35. J.H. Grace and A. Young. *The Algebra of Invariants*. Cambridge University Press, 1903.
36. A. Heck. A bird's-eye view of Gröbner bases. In *Proceedings of 5th* AIHENP *(International Workshop on New Computing Techniques in Physics Research)*, 1996.
37. G. Helzer. Gröbner bases. *Mathematical Journal*, 5(1):67–73, 1995.
38. C. Hermite. Remarques sur le théorème de Sturm. *Comptes-Rendus de l'Académie des Sciences de Paris*, 36:52–54, 1853.
39. D. Hilbert. *Theory of Algebraic Invariants*. Cambridge Mathematica Library, Cambridge University Press, 1890.
40. D. Hilbert. Über die theorie der algebraischen formen. *Math. Annalen*, 36:473–534, 1890.
41. C. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers Inc., 1989.
42. B. Huber and B. Sturmfels. A polyhedral method for solving sparse polynomial systems. *Math. Comp.*, 64:1541–1555, 1995.
43. B. Huber and B. Sturmfels. Bernstein's theorem in affine space. *Discrete and Computational Geometry*, 1996. To appear.
44. I. Itenberg and M.-F. Roy. Multivariate Descartes' rule. *Beiträge zur Algebra und Geometrie*, 37(2):337–346, 1996.
45. A.K. Jain and P.J. Flynn, editors. *Three-Dimensional Object Recognition Systems*, volume 1 of *Advances in Image Communication*. Elsevier Press, 1993.
46. M. Jirstrand. Cylindrical algebraic decomposition - an introduction. Technical Report LiTH-ISY-R-1807, Department of Electrical Engineering, Linköping University, 1995.
47. J.-P. Jouanolou. Le formalisme du résultant. *Advances in Mathematics*, 98:117–263, 1991.
48. M. Kac. On the average number of real roots of a random algebraic equation. *Bulletin of the American Mathematical Society*, 49:314–320, 1943.
49. D. Kapur and Y.N. Lakshman. Elimination methods: an introduction. In B.R. Donald, D. Kapur, and J.L. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence*, pages 45–89. Academic Press, 1992.
50. A.G. Khovanskii. Fewnomials. In *Translations of Mathematical Monographs*, volume 88. American Mathematical Society, 1991.
51. F. Klein. Eine neue Relation zwischen den Singularitäten einer algebraischen Curve. *Math. Annalen*, 10:199–209, 1876.
52. J.J. Koenderink and A.J. van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.
53. E. Kostlan. On the distribution of the roots of random polynomials. In M. Hirsch, J. Marsden, and M. Shub, editors, *From Topology to Computation*. Springer-Verlag, 1991. Proceedings of the Smalefest.
54. D. Lazard. Résolution des systèmes d'équations algébriques. *Theoretical Computer Science*, 15:77–110, 1981.
55. T. Li and X. Wang. The BKK root count in $\mathbb{C}^n$. *Mathematics of Computation*, 65(216):1477–1484, October 1996.
56. G. Lyubeznik. Minimal resultant systems. *Journal of Algebra*, 177:612–616, 1995.
57. F.S. Macaulay. Some formulae in elimination. *Proceedings of the London Mathematical Society*, 1(33):3–27, 1902.
58. F.S. Macaulay. *The Algebraic Theory of Modular Systems*, volume 8 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1916.
59. D. Manocha and J.F. Canny. The implicit representation of rational parametric surfaces. *Journal of Symbolic Computation*, 13:485–510, 1992.
60. M. Mignotte. Some useful bounds. *Computer Algebra: Symbolic and Algebraic Computation*, pages 259–263, 1982.
61. J. Milnor. On the Betti numbers of real varieties. In *Proceedings of the American Mathematical Society*, volume 15, pages 275–280, 1964.
62. B. Mishra. *Algorithmic Algebra*. Springer-Verlag, 1993.
63. A.P. Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems*. Prentice Hall, Englewood Cliffs, New Jersey, 1987.
64. B. Mourrain. Solving polynomial systems by matrix computations, 1997. Preprint.

65. B. Mourrain and N. Stolfi. Computational symbolic geometry. In *Invariant Methods in Discrete and Computational Geometry*, pages 107–139. Kluwer Academic Publishers, 1994.
66. J.L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.
67. V.Y. Pan, J.H. Reif, and S.R. Tate. The power of combining the techniques of algebraic and numerical computing: improved approximate multipoint polynomial evaluation and improved multipole algorithms. In *Proceedings of* FOCS*'92 (*IEEE *Symposium on Foundations of Computer Science)*, pages 703–713, 1992.
68. P. Pedersen, M.-F. Roy, and A. Szpirglas. Counting real zeros in the multivariate case. In F. Eyssette and A. Galligo, editors, *Proceedings of* MEGA*'92 (International Symposium on Effective Methods in Algebraic Geometry)*, Progress in Mathematics, pages 203–224, Nice, France, 1992. Birkhäuser.
69. S. Petitjean. The enumerative geometry of projective algebraic surfaces and the complexity of aspect graphs. *International Journal of Computer Vision*, 19(3):1–27, 1996.
70. S. Petitjean, J. Rieger, and D. Forsyth. Recognizing algebraic surfaces from aspects. In J. Ponce, editor, *Algebraic Surfaces in Computer Vision*. Springer-Verlag, 1997. Series in Information Sciences, To appear.
71. A. Rege. A complete and practical algorithm for geometric theorem proving. In *Proceedings of 11th* SCG *(*ACM *Annual Symposium on Computational Geometry), Vancouver, Canada*, pages 277–286, 1995.
72. A. Rege. *A Toolkit for Algebra and Geometry*. PhD thesis, Computer Science Department, University of California at Berkeley, 1996.
73. J. Renegar. On the computational complexity and geometry of the first-order theory of the reals I, II and III. *Journal of Symbolic Computation*, 13(3):255–300, 301–328, 329–330, 1992.
74. J.H. Rieger. Computing view graphs of algebraic surfaces. *Journal of Symbolic Computation*, 16:259–272, 1993.
75. J.H. Rieger. Notes on the complexity of exact view graph algorithms for piecewise-smooth algebraic surfaces. Preprint. Universidade de Sao Paulo, Brazil, 1996.
76. J.H. Rieger. On the complexity and computation of view graphs of piecewise-smooth algebraic surfaces. *Philosophical Transactions of the Royal Society of London, Series A*, 354(1714):1899–1940, 1996.
77. M. Rojas. A convex geometric approach to counting the roots of a polynomial system. *Theoretical Computer Science*, 133(1):105–140, 1994.
78. M. Rojas. On the average number of real roots of certain random sparse polynomial systems. In Jim Renegar, Mike Shub, and Steve Smale, editors, *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Mathematics*, pages 689–699. American Mathematical Society, 1996.
79. M. Rojas. Toric generalized characteristic polynomials, 1997. Submitted to ISSAC (International Symposium on Symbolic and Algebraic Computation).
80. M. Rojas. Toric intersection theory for affine root counting. *Journal of Pure and Applied Algebra*, 1997. To appear.
81. M. Rojas and X. Wang. Counting affine roots via pointed Newton polytopes. *Journal of Complexity*, 12:116–133, 196.
82. F. Ronga. Felix Klein's paper on real flexes vindicated. Université de Genève, preprint, September 1996.
83. F. Ronga, A. Tognoli, and T. Vust. The number of conics tangent to 5 given conics: the real case. Université de Genève, preprint, March 1995.
84. F. Ronga and T. Vust. Stewart platforms without computer? In Walter de Gruyter, editor, *Proceedings of the 1992 International Conference on Real Analytic and Algebraic Geometry*, pages 196–212, Trento, 1995.
85. F. Rouillier. *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux*. PhD thesis, Université de Rennes I, 1996.
86. M.-F. Roy, L. Gonzalez, H. Lombardi, and T. Recio. Sturm-Habicht, determinants, and real roots of univariate polynomials. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer-Verlag, 1994.
87. M.-F. Roy and T. van Effelterre. Aspect graphs of bodies of revolution with algorithms of real algebraic geometry. *Progress in Mathematics*, 143, 1996. Proceedings of MEGA'94 (International Symposium on Effective Methods in Algebraic Geometry), Santander, Spain, April 1994.
88. A. Salden. *Dynamic Scale-Space Paradigms*. PhD thesis, Utrecht University, The Netherlands, 1996.
89. T.W. Sederberg and F. Chen. Implicitization using moving curves and surfaces. *Computer Graphics Proceedings, Annual Conference Series*, 29:301–308, 1995. Proceedings of SIGGRAPH'95.
90. M. Sharir and P.K. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications*. Cambridge University Press, 1995.
91. M. Shub and S. Smale. Complexity of Bezout's theorem I: geometric aspects. *Journal of the American Mathematical Society*, 6:459–501, 1993.
92. M. Shub and S. Smale. Complexity of Bezout's theorem II: volumes and probabilities. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 267–285. Birkhäuser, 1993.
93. M. Shub and S. Smale. Complexity of Bezout's theorem V: polynomial time. *Theoretical Computer Science*, 133:141–164, 1994.
94. M. Shub and S. Smale. Complexity of Bezout's theorem IV: probability of success, extensions. SIAM *Journal of Numerical Analysis*, 33(1):128–148, 1996.
95. F. Sottile. Enumerative geometry for real varieties. Technical report, Mathematical Sciences Research Institute, 1996. AMS preprint #199609-14-009.
96. F. Sottile. Enumerative geometry for the real Grassmannian of lines in projective space. *Duke Mathematical Journal*, 1996. To appear.
97. F. Sottile. Real enumerative geometry and effective algebraic equivalence. In *Proceedings of 4th* MEGA *(International Symposium on Effective Methods in Algebraic Geometry), Eindhoven, The Netherlands*, 1996.

98. P. Stiller. An introduction to the theory of resultants. Technical Report ISC-96-02-MATH, Texas A & M University, Institute for Scientific Computation, 1996.
99. P. Stiller. Sparse resultants. Technical Report ISC-96-01-MATH, Texas A & M University, Institute for Scientific Computation, 1996.
100. P. Stiller, C. Asmuth, and C. Wan. Invariants, indexing, and single view recognition. In *Proceedings of* IUW *(Image Understanding Workshop)*, pages 1423–1428, 1994.
101. C. Sturm. Mémoire sur la résolution des équations numériques. *Inst. France Sc. Math. Phys.*, 6, 1835.
102. B. Sturmfels. On the number of real roots of a sparse polynomial system. In A. Bloch, editor, *Hamiltonian and Gradient Flows: Algorithms and Control*, volume 3, pages 137–143. American Mathematical Society, 1991.
103. B. Sturmfels. Introduction to resultants. In *Proceedings of the* AMS *short course on Applications of Computational Geometry, San Diego*, 1997. January 6-7.
104. B. Sturmfels and A. Zelevinsky. Multigraded resultants of Sylvester type. *Journal of Algebra*, 163(1):115–127, 1994.
105. J.J. Sylvester. On a general method of determining by mere inspection the derivations from two equations of any degree. *Philosophical Magazine*, 16:132–135, 1840.
106. A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.
107. B.L. van der Waerden. *Modern Algebra*. Frederick Ungar Publishing Co., 1970. Fifth Edition.
108. T. van Effelterre. *Calcul exact du graphe d'aspect de solides de révolution*. PhD thesis, Université de Rennes I, 1995.
109. L. Velho and J. Gomes. Approximate conversion from parametric to implicit surfaces. In B. Wyvill and M.-P. Gascuel, editors, *Proceedings of First International Workshop on Implicit Surfaces*, pages 77–96, Grenoble, France, April 1995.
110. V. Weispfenning. Solving parametric polynomial equations and inequalities by symbolic algorithms. In J. Fleischer, J. Grabmeier, F. W. Hehl, and W. Küchlin, editors, *Computer Algebra in Science and Engineering*. World Scientific Publishing, 1995.
111. R. Weitzenböck. *Invariantentheorie*. P. Noordhoff, Groningen, 1923.
112. M. Werman and A. Shashua. The study of 3D-from-2D using elimination. In *Proceedings of 5th* ICCV *(International Conference on Computer Vision), Cambridge, Massachusetts*, pages 473–479, 1995.
113. J. Weyman and A. Zelevinsky. Determinantal formulas for multigraded resultants. *Journal of Algebraic Geometry*, 3:569–597, 1994.
114. W.-T. Wu. On the decision problem and the mechanization of theorem-proving in elementary geometry. In W. Bledsoe and D. Loveland, editors, *Automated Theorem Proving: After 25 Years*, pages 213–234. American Mathematical Society, 1983. Contemporary Mathematics 29.
115. B. Wyvill and M.-P. Gascuel, editors. *Proceedings of Implicit Surfaces'95 (First International Workshop on Implicit Surfaces)*, Grenoble, France, April 1995.
116. H.G. Zeuthen. Abzählende Methoden. In *Enzyklopädie der Mathematischen Wissenschaften*, pages 43–87. Leipzig, 1903–1915. dritter Band, zweiter Teil, erste Hälfte.