Real solving for positive dimensional systems

P. Aubry*, F. Rouillier†, M. Safey El Din*

March 14, 2000

Abstract

Finding one point on each semi-algebraically connected component of a real algebraic variety, or at least deciding if such a variety is empty or not, is a fundamental problem of computational real algebraic geometry. Even though numerous studies have been done on the subject, only a few number of efficient implementations exists.

In this paper, we propose a new efficient and practical algorithm for computing such points. By studying the critical points of the restriction to the variety of the distance function to one well chosen point, we show how to provide a set of zero-dimensional systems whose zeroes contain at least one point on each semi-algebraically connected component of the studied variety, without any assumption neither on the variety (smoothness or compactness for example) nor on the system of equations that define it.

Once such a result is computed, one can then apply, for each computed zero-dimensional system, any symbolic or numerical algorithm for counting or approximating the solutions. We have made experiments using a set of pure exact methods.

The practical efficiency of our method is due to the fact that we do not apply any infinitesimal deformations, conversely to the existing methods based on similar strategy.

1 Introduction

The problem of finding one point on each semi-algebraically connected component of a real algebraic variety, or at least deciding if it is empty, appears in several computational problems in computational algebraic geometry.

The most popular algorithm which solves this problem is Collins' Cylindrical Algebraic Decomposition (see [8]). This algorithm is based on variable elimination, one after the other, and solves the thruth decisison problem of a first order formula. Thus, it solves more general problems than the one in which we are interested. Note also that it is polynomial in the degree and the number of the polynomials and doubly exponential in the number of variables. In practice, this theoretical complexity can be observed for many examples, and so, the problem size which can be solved with such algorithms is limited.

In [14], Grigoriev and Vorobjov proposed an algorithm for deciding the emptiness of a semi-algebraic set with a single exponential complexity in the number of variables. In this method as well as in most of its variants (see [20, 7, 15, 4, 5, 25]), the key idea is to apply deformations so that the projection critical points with respect to one coordinate define a finite set that meets every semi-algebraic connected component of the deformed variety. In [4, 5, 25] the authors take, in

*Université de Paris VI, France

†LORIA, INRIA-Lorraine, Nancy, France

addition, sums of squares in order to work with smooth and compact real algebraic sets defined by a unique polynomial equation. The final result is then obtained by taking the limits of the points (substituting the infinitesimals by zero).

In any case, the problem is reduced to the resolution of zero-dimensional systems. But even if the various done transformations keep a good theoretical complexity (see [4]), they render impossible an efficient resolution in practice, due to the use of at least two infinitesimals (deformations) and a degree growth (sum of squares).

In [3] , the authors provide an algorithm, based on straight-line programs, with a good theoretical complexity, when the variety to be studied is smooth, compact and given by a regular sequence of polynomials, so that, in practice, one would have to face, at least, the same problems than in [4] (smoothness and compactness) for providing an algorithm that works in every situation.

In [9], the authors propose a new algorithm for deciding the emptiness of semi-algebraic sets, which is more practical. In particular they avoid to take the sums of the squares, and deal with the singularities by using the fact that the singular locus is an algebraic variety whose dimension is inferior to the one of the variety they consider. Nevertheless, they keep on using the projection function. Thus, their algorithm requires the use of at least one infinitesimal deformation or a new variable addition for dealing with non compact varieties.

In [23], the authors consider the particular case of a variety defined by a single equation. Coming back to a classical idea of Seidenberg (see [26]), they study the critical points of the distance function to a point instead of coordinates functions. The authors recall that the set of critical point set of the distance function to a point meet each connected component and they show that it is finite when the point is well chosen (they propose a strategy for choosing it) and when the variety has at most a finite number of singularities, so that an infinitesimal deformation is needed only when the variety has an infinite number of singular points.

In this paper, we keep on computing the critical points of the distance function to a point A but for the general case of real algebraic sets defined by a polynomial system of equations (the case of hypersurfaces defined by a unique equation becomes a particular case without taking the sum of the squares of the equations). Like in [23], we define an algebraic set $\mathcal{C}(V, A)$ that contains these critical points and a sub-algebraic variety of V of the one studied.

Our main result consists in proving that by choosing a good point A, $\mathcal{C}(V, A)$ is the disjoint union of a sub-algebraic variety of V whose dimension is inferior to the one of the variety and of a finite set of points. The problem then remains to compute the isolated points of $\mathcal{C}(V, A)$ and to study, in the same way, the defined sub-variety V which has a dimension strictly smaller than the dimension of the variety. Also, we obtain an algorithm without any infinitesimal deformation whose proof is simply based on the fact that the dimension of the studied varieties strictly decreases at each step.

The paper plan is as follows. Section 2 is devoted to the definition and the study of $\mathcal{C}(V, A)$, an algebraic set depending on the choice of a point A in the space, that meets every semi-algebraically connected component of V. In particular, we show how to choose a point A so that $\mathcal{C}(V, A)$ becomes the disjoint union of a finite set of points and of V. Moreover we propose the explicit construction of a zero-dimensional system whose zeroes meet every semi-algebraic connected component of V. Section 3 is devoted to present the way of using in practice such results for getting an efficient algorithm that avoids infinitesimal deformations. The last section is devoted to present some practical experiments.

Acknowledgments : We would like to thank J.-C. Faugère, D. Lazard and M.-F. Roy for their helpful comments, advises and supports and Hoon Hong who did provide us the CAD implementation

used for the tests.

2 The Algorithm

In the whole paper, K is an ordered field, R is its real closure and C its algebraic closure. If $(P_1, \ldots, P_s)$ is a family of polynomials in $K[X_1, \ldots, X_n]$, we denote by $V(P_1, \ldots, P_s) \subset C^n$ the algebraic variety defined by the polynomial system of equation :

$$P_1 = \ldots = P_s = 0$$

and $I = \langle P_1, \ldots, P_s \rangle$ the ideal of $K[X_1, \ldots, X_n]$ generated by this family of polynomials.

As described in introduction, our goal is to use the properties of the distance function to one point. More precisely, we are going to prove the following theorem :

Theorem 2.1 *Let V be an algebraic variety of dimension d and $S = \{P_1, \ldots, P_s\}$ polynomials of $K[X_1, \ldots, X_n]$ such that $I(V) = \langle P_1, \ldots, P_s \rangle$. Given any point $A \in C^n$, we define the following algebraic set :*

$$\mathcal{C}(V, A) = \{M \in V, rank(\overrightarrow{grad}_M(P_1), \ldots, \overrightarrow{grad}_M(P_s), \overrightarrow{AM}) \leq n - d\}.$$

If D is a positive integer large enough, there exists at least one point A in $\{1 \ldots D\}^n$ such that :

1. *$\mathcal{C}(V, A)$ meets every semi-algebraically connected component of $V \bigcap R^n$,*
2. *$\mathcal{C}(V, A) = \mathrm{Sing}(V) \bigcup V_0$.*

where

- *V_0 is a finite set of points in C^n,*
- $\mathrm{Sing}(V) = \{M \in V \mid \mathrm{rank}(\overrightarrow{grad}_M(P_1), \ldots, \overrightarrow{grad}_M(P_s)) < n - d\}$.

Moreover, $dim(\mathcal{C}(V, A)) < dim(V)$.

Proof : Let A be any point in C^n and $\mathcal{D}$ be a semi-algebraically connected component of $V \bigcap R^n$. If $M \in \mathcal{D} \bigcap \mathrm{Sing}(V)$, it is clear that $M \in \mathcal{C}(V, A)$.
Now, suppose that $M \in \mathcal{D} \setminus \mathrm{Sing}(V)$ is at minimal distance from A. Let $\mathcal{S}(A, r)$ be the sphere of center A and radius $r = d(A, M)$. Since M is at minimal distance to A, $\mathcal{S}$ and V are tangent at M and then $\overrightarrow{AM} \in \mathrm{Vect}(\overrightarrow{grad}_M(P_1), \ldots, \overrightarrow{grad}_M(P_s))$. Thus $M \in \mathcal{C}(V, A)$.
Let $Q_1, \ldots, Q_n$ be polynomials in $K[X_1, \ldots, X_n, \lambda_1, \ldots, \lambda_s]$ defined by $Q_j = \sum_{i=1,\ldots,s} \lambda_i \frac{\partial P_i}{\partial X_j} - X_j$, and let $\mathcal{H}$ be the subset of C^{n+s} defined by $\mathcal{H} = \{(M, \lambda_1, \ldots, \lambda_s) \in C^{n+s} \mid M \in V \setminus \mathrm{Sing}(V)\}$.
Consider the application

$$\begin{array}{rccc} F : & \mathcal{H} & \longrightarrow & C^n \\ & (M, \lambda_1, \ldots, \lambda_s) & \longmapsto & (Q_1(M, \lambda_1, \ldots, \lambda_s), \ldots, Q_n(M, \lambda_1, \ldots, \lambda_s)) \end{array}$$

If $Jac((P_1, \ldots, P_s, Q_1 + b_1, \ldots, Q_n + b_n)$ denotes the determinant of the Jacobian matrix associated to the polynomials $P_1, \ldots, P_s, Q_1 + b_1, \ldots, Q_n + b_n$, the critical values of F are the points $B = (b_1, \ldots, b_n)$ of C^n such that$V(Q_1 + b_1, \ldots, Q_n + b_n, \mathrm{Jac}(P_1, \ldots, P_s, Q_1 + b_1, \ldots, Q_n + b_n)) \neq \emptyset$.

¿From Sard's theorem over C [19] and the transfer principle [6] it follows that

$$\mathcal{B} = \{B = (b_1, \ldots, b_n) \in C^n \mid$$
$$\mathcal{H} \cap V(Q_1 + b_1, \ldots, Q_n + b_n, \mathrm{Jac}(P_1, \ldots, P_s, Q_1 + b_1, \ldots, Q_n + b_n)) \neq \emptyset\}$$

is a constructible set of dimension $< n$ of C^n.
Since $\mathcal{B}$ is a constructible set of dimension $< n$, one can choose $A = (a_1, \ldots, a_n) \in \{0, \ldots, D\}^n$ with D large enough, and such that $A \notin \mathcal{B}$. In such case,

$$\mathcal{H} \cap V(Q_1 + a_1, \ldots, Q_n + a_n, \mathrm{Jac}(P_1, \ldots, P_s, Q_1 + a_1, \ldots, Q_n + a_n)) = \emptyset$$

and thus the points of $\mathcal{H} \cap V(Q_1 + a_1, \ldots, Q_n + a_n)$ are isolated and non singular. Let π be the projection defined by :

$$\begin{array}{rccc} \pi : & C^{n+s} & \longrightarrow & C^n \\ & (x_1, \ldots, x_n, \ell_1, \ldots, \ell_s) & \longmapsto & (x_1, \ldots, x_n) \end{array} .$$

Since $\mathcal{C}(V, A) = \mathrm{Sing}(V) \cup \pi(\mathcal{H} \cap V(Q_1 + a_1, \ldots, Q_n + a_n))$, $\mathcal{C}(V, A) = \mathrm{Sing}(V) \cup V_0$, where V_0 is finite set of points. ¿From [10], $\mathrm{Sing}(V)$ is the union of algebraic varieties whose dimensions are strictly inferior to the dimension of V. ■

Remark 2.1 *¿From the proof of theorem 2.1, a point A taken at random verifies* $\dim(\mathcal{C}(V, A)) < \dim(V)$ *with a probability one.*

Definition 2.1 *Given any variety $V \in C^n$, we define recursively $\mathcal{C}^i(V, A_i)$, $i \geq 0$, a set of points in C^n in the following way :*

- *A_0 is any point in K^n and $\mathcal{C}^0(V, A_0) = V$,*
- *$A_i \in K^n$ is such that* $\dim(\mathcal{C}(\mathcal{C}^{i-1}(V, A_{i-1}), A_i)) < \dim(\mathcal{C}^{i-1}(V, A_{i-1}))$,
- *$\mathcal{C}^i(V, A_i) = \mathcal{C}(\mathcal{C}^{i-1}(V, A_{i-1}), A_i)$.*

According to the definition above, we have :

Corollary 2.1 *Given any variety $V \in C^n$, It exists an integer $m < \infty$ such that :*

- *$\mathcal{C}^m(V, A_m)$ is a finite set of points,*
- *$\mathcal{C}^m(V, A_m))$ meets every semi-algebraically connected component of $V \bigcap R^n$.*

Proof : ¿From theorem 2.1 , $\mathcal{C}^{i+1}(V, A_{i+1})$ meets every semi-algebraically connected component of $\mathcal{C}^i(V, A_i) \bigcap R^n$ and $\dim(\mathcal{C}^{i+1}(V, A_{i+1})) < \dim(\mathcal{C}^i(V, A_i))$, $\forall i \geq 0$. The proof comes also immediately by induction since $\mathcal{C}^0(V, A_0) = V$ and since $\forall i \in \{0, \ldots, m\}$ $V(C^{i+1}(V, A_{i+1})) \subset V(C^i(V, A_i))$. ■

The algorithm we propose consists in constructing the sets $\mathcal{C}^i(V, A_i)$ until $\dim(\mathcal{C}^m(V, A_m)) = 0$. In a computational viewpoint, this consists in computing a set of generators $\mathcal{P}_i$ of radical ideals $\mathcal{I}_i$ such that $V(\mathcal{I}_i) = \mathcal{C}^i(V, A_i)$, $\forall i = 0 \ldots m$.

Let suppose that $\mathcal{P}_k = \{P_{k,1}, \ldots P_{k,s}\}$ such a set.

Definition 2.2 *For $B \in C^n$, $\mathcal{Q} = \{Q_1, \ldots Q_s\} \subset K[X_1, \ldots, X_n]^s$, and $d \in \mathbb{N}$, $0 \leq d < n$, we define $\Delta_{B,d}(\mathcal{Q})$ as being the set of all the minors of order $(n-d+1, n-d+1)$ of the matrix*

$$\left[\left[\frac{\partial Q_i}{\partial X_j} \right]_{(i=1\ldots n, j=1\ldots s)} \middle| \overrightarrow{BM} \right]$$

Coming back to our problem, if $V(\langle \mathcal{P}_i \rangle) = \mathcal{C}^i(V, A_i)$, $d_i = \dim(\mathcal{C}^i(V, A_i)$ and $\langle \mathcal{P}_i \rangle$ is a radical ideal, then

$$\mathcal{C}^{i+1}(V, A_{i+1}) = V(\langle \mathcal{P}_i \ , \ \Delta_{A_{i+1}, d_i}(\mathcal{P}_i) \rangle).$$

We may also define $\mathcal{P}_{i+1}$ as being a system of generators of the ideal $\sqrt{\langle \mathcal{P}_i \Delta_{A_{i+1}, d_i}(\mathcal{P}_i) \rangle}$.
According to the results above, the basic routines needed to implement an algorithm that computes a zero-dimensional system $\mathcal{P}_m$ such that $V(\mathcal{P}_m) = \mathcal{C}^m(V, A_m)$ (corollary 2.1) may be the following :

- **Radical** : takes as input a polynomial system S of equations and returns a finite set of generators (for example a Gröbner base) of $\sqrt{\langle S \rangle}$,
- **Dim** : takes as input a finite set of generators of an ideal and computes the dimension of the associated variety,
- **Minors** : takes as input a finite set of polynomials $\mathcal{Q}$, and integer d and a point $A \in C^n$ (in fact in K^n) and computes $\Delta_{A,d}(\mathcal{Q}) \rangle)$.

Algorithm 1

- **Input :** A polynomial system S of equations in $K[X_1, \ldots, X_n]$.
- **Output :** A zero-dimensional system whose zeroes define at least one point in each semi-algebraically connected component of $V(S) \bigcap R^n$.

1. $S :=$ Radical(S), $d :=$ Dim(S),
2. Choose $A \notin V(S)$.
3. while $d \neq 0$ do
   - (*) $Q =$ Minors$(S, d, A) \bigcup S$
   - $u =$ Dim(Q)
   - if $u = d$ choose another point A and go to step (*).
   - else $d := u$; $S :=$ Radical(Q)
4. return (S).

Note that the required subroutines of our algorithm are weaker than the ones of the algorithm described in [9] since we do not need to perform an irreducible decomposition.

3 Optimizations

In this section, we present practical optimizations of our algorithm. The main idea is to split, as most as possible, the systems to be solved ($\sqrt{\langle \mathcal{P}_i \rangle}$) in order to make easier the intermediate computations and the resolution of the final zero-dimensional systems. In the first part, we modify

slightly **Algorithm 1** for this purpose. In the second part, we show how to decrease the number of determinants to compute and to make their computation easier. The last sub-section is devoted to present a lazy version of the algorithm for deciding the emptiness of a real algebraic variety, and computing at least one point on each semi-algebraically connected component in some *generic* cases.

In the two last parts, we use results coming from the theory of *triangular sets* (see [17, 18, 16, 1, 2]) for optimizing the algorithms. We refer to [2] which synthesizes these results.

3.1 Splitting the computations

In **Algorithm 1**, we need to compute generators of radical ideals at each step. According to recent progress (see [11]), the most efficient way to do such a computation in practice is to compute a decomposition into primes. Moreover, since each computed system splits into a zero-dimensional system and a system of dimension less than the original one (theorem 2.1), we may extract the isolated roots at each step instead of keeping them embedded in the main component, making easier the final resolution (for example isolating the real roots of zero-dimensional systems).

Let **PrimeDecomposition** be a subroutine taking as input a polynomial system of equations S in $K[X_1, \ldots, X_n]$ and returning a set of generators of each prime ideal associated to $\sqrt{\langle S \rangle}$.

We propose the following algorithm :

Algorithm 2

- **Input :** A polynomial system S of equations in $K[X_1, \ldots, X_n]$.
- **Output :** An empty list if $V(S) \bigcap R^n = \emptyset$, else a list of zero-dimensional systems whose roots contain at least one point in each semi-algebraically connected component of $V(S) \bigcap R^n$.

1. list := PrimeDecomposition(S), result := [],
2. Choose $A \notin V(S)$.
3. while list $\neq \emptyset$ do
   - S := first(list), and remove S from list, set $d = \mathrm{Dim}(S)$,
   - if $d = 0$ then result := result $\bigcup S$,
   - else
     - (*) $Q = \mathrm{Minors}(S, d, A) \bigcup S$ and set $u = \mathrm{Dim}(Q)$
     - if $u = d$ choose another point $A \notin S$ and go to step (*).
     - $d := u$; list := list $\bigcup$ PrimeDecomposition(Q),
4. return result.

3.2 Using triangular sets

In this section, we show how to reduce the number and the size of the determinants by considering specific subsets of the Gröbner bases.

Let $\mathcal{G}$ be a reduced lexicographical Gröbner base generating a prime ideal of dimension d in $K[X_1, \ldots, X_n]$ for the ordering $X_1 < \ldots < X_n$. For $p \in \mathcal{G}$, we denote by $\mathrm{mvar}(p)$ (and we call main variable of p) the greatest variable appearing in p . If F is a constructible subset of C^n, we denote by $\overline{F}$ the Zariski closure of F in C^n. In [2], the following result is proved :

Theorem 3.1 *(see [2] and [1]) Let $\mathcal{T} = (t_{d+1}, \ldots, t_n) \subset \mathcal{G}$ be a set of polynomials such that*

$$\forall (t_i, t_j) \in \mathcal{T} \times \mathcal{T} \ \mathrm{mvar}(t_i) \neq \mathrm{mvar}(t_j),$$

and $\forall g \in \mathcal{G},\ \forall i \in \{d+1,\ldots,n\}$ *such that* $\mathrm{mvar}(t_i) = \mathrm{mvar}(g)$ *([1]) :*

$$\deg(t_i, \mathrm{mvar}(t_i)) \leq \deg(g, \mathrm{mvar}(t_i)).$$

We denote by

- h_i *the leading coefficient of* t_i *(when it is seen as a univariate polynomial in its main variable) and* $\mathcal{H}(\mathcal{T}) = \{h_{d+1}, \ldots, h_n\}$.
- $W(\mathcal{T}) = \{M \in V(\mathcal{T}) \setminus V(\mathcal{H})\}$,
- $\mathrm{sat}(\mathcal{T}) = \{p \in K[X_1, \ldots, X_n] \mid \exists m \in N, \exists h \in \mathcal{H}(\mathcal{T}), h^m p \in \langle \mathcal{T} \rangle\}$.

Then we have :

1. $\mathrm{sat}(\mathcal{T}) = \langle \mathcal{G} \rangle$,
2. $\overline{W(\mathcal{T})} = V(\mathcal{G})$,

If $\mathcal{G}$ is prime, the set $\mathcal{T} = (t_{d+1}, \ldots, t_n)$ is a regular separable triangular set (see [2]).

Lemma 3.1 *If* $\mathcal{T}$ *is a regular separable triangular set extracted from a reduced lexicographical Gröbner base generating a prime ideal, then :* $\forall i \in \{d+1,\ldots,n\}, \dim(V(\frac{\partial t_i}{\partial \mathrm{mvar}(t_i)}) \bigcap V(\mathcal{G})) < \dim(V(\mathcal{G}))$.

Proof : Since $\langle \mathcal{G} \rangle$ is prime, it is sufficient to show that $\frac{\partial t_i}{\partial \mathrm{mvar}(t_i)} \notin \langle \mathcal{G} \rangle$. Let $mX_i^{d_i}$ the leading monomial of t_i. Since $\mathcal{G}$ is reduced, $\forall g \in \mathcal{G} \setminus \{t_i\}$, the leading monomial of g does not divide $mX_i^{d_i}$. Hence, the leading monomial of $\frac{\partial t_i}{\partial \mathrm{mvar}(t_i)}$ is not divisible by the leading monomial of g, and thus $\frac{\partial t_i}{\partial \mathrm{mvar}(t_i)} \notin \langle \mathcal{G} \rangle$. ■

Let $M = (x_1, \ldots, x_n)$, $A = (a_1, \ldots a_n)$, $d = \dim(V(\mathcal{G}))$, and consider, for $j = 1 \ldots d$, the restricted list of minors of order $(n-d+1)$ extracted from $\Delta_{A,d}(\mathcal{T})$:

$$\Gamma_A(\mathcal{T}) = \{\Gamma_A^{(j)}(\mathcal{T}) = det(\mathcal{M}_A^{(j)}), j = 1 \ldots d\}$$

where

$$\mathcal{M}_A^{(j)} = \left[\begin{array}{c|c} \left[\frac{\partial t_i}{\partial X_j}\right]_{i=d+1\ldots n} & x_j - a_j \\ \hline \mathcal{U}_{\mathcal{T}} = \left[\frac{\partial t_i}{\partial X_j}\right]_{j=d+1\ldots n}^{i=d+1\ldots n} & \begin{array}{c} x_{d+1} - a_{d+1} \\ \vdots \\ x_n - a_n \end{array} \end{array}\right]$$

Without lost of generality, we may suppose that $\mathrm{mvar}(t_i) = X_i$, so that the minors $\Gamma_A^{(j)}(\mathcal{T})$ are easy to compute since $\mathcal{U}_{\mathcal{T}}$ is upper triangular. Our goal is now to show that we can replace, in our algorithm, the computation of $\Delta_{A,d}(\mathcal{G})$ by the computation of $\Gamma_A(\mathcal{T})$, decreasing so the number and the cost of the computations :

Proposition 3.1 *Let define* $\mathcal{D}(V(\mathcal{G}), A) = V(\mathcal{G}) \bigcap V(\Gamma_A(\mathcal{T}))$, $d = dim(\mathcal{G})$ *and* $Sep(\mathcal{T}) = \prod_{i=d+1}^{n} \frac{\partial t_i}{X_i}$. *If* $A \in C^n$ *such that* $dim(\mathcal{C}(V(\mathcal{G}), A)) < dim(V(\mathcal{G}))$, *then, according to the notations of theorem 2.1, we have :*

- $\mathcal{C}(V(\mathcal{G}), A) \subset \mathcal{D}(V(\mathcal{G}), A)$,
- $(\mathcal{D}(V(\mathcal{G}), A) \setminus V(Sep(\mathcal{T}))) \subset V_0$,
- $dim\,(\mathcal{D}(V(\mathcal{G}), A) \bigcap V(Sep(\mathcal{T}))) < dim(V(\mathcal{G}))$.

In particular, $dim(\mathcal{D}(V(\mathcal{G}), A)) < dim(V(\mathcal{G}))$. *and* $\mathcal{D}(V(\mathcal{G}), A)$ *meets every semi-algebraically connected component of* $V(\mathcal{G})$.

Proof :

- Since $\mathcal{T} \subset \mathcal{G}$, $\Gamma_A(\mathcal{T}) \subset \Delta_{A,d}(\mathcal{T}) \subset \Delta_{A,d}(\mathcal{G})$, then :

$$\mathcal{C}(V(\mathcal{G}), A) = V(\mathcal{G}) \bigcap V(\Delta_{A,d}(\mathcal{G})) \subset V(\mathcal{G}) \bigcap V(\Delta_{A,d}(\mathcal{T})) \subset V(\mathcal{G}) \bigcap V(\Gamma_A(\mathcal{T})).$$

- Let $M \in \mathcal{D}(V(\mathcal{G}), A) \setminus V(Sep(\mathcal{T}))$. We have $det(\mathcal{U}_{\mathcal{T}}(M)) \neq 0$ so that

$$\text{rank}(\overrightarrow{grad}_M(t_{d+1}), \ldots, \overrightarrow{grad}_M(t_n)) \geq n - d,$$

  and consequently $\text{rank}(\overrightarrow{grad}_M(g_1), \ldots, \overrightarrow{grad}_M(g_s)) \geq n - d$. On one hand, $\Gamma_A^{(i)}(\mathcal{T})(M) = 0$, $\forall i = 1 \ldots d$, and so $rank(\overrightarrow{grad}_M(t_{d+1}), \ldots, \overrightarrow{grad}_M(t_n), \overrightarrow{AM}) = n - d$. On the other hand, $dim(V(\mathcal{G})) = d$, so that $\text{rank}(\overrightarrow{grad}_N(g_1), \ldots, \overrightarrow{grad}_N(g_s)) \leq n - d$, $\forall N \in V(\mathcal{G})$ and thus $M \notin Sing(V(\mathcal{G}))$. Moreover, the vector spaces $Vect(\overrightarrow{grad}_M(g_1), \ldots, \overrightarrow{grad}_M(g_s))$ and $Vect(\overrightarrow{grad}_M(t_{d+1}), \ldots, \overrightarrow{grad}_M(t_n))$ coincide which shows that $M \in V_0 = \mathcal{C}(V(\mathcal{G})) \setminus Sing(V(\mathcal{G}))$.
- From 3.1, $\dim(V(Sep(\mathcal{T})) \bigcap V(\mathcal{G})) < \dim(V(\mathcal{G}))$.

■

For describing the full algorithm induced by proposition 3.1, we define new external functions :

- **LexPrimeDecomposition** takes as input a polynomial system of equations S in $K[X_1, \ldots, X_n]$ and returns a lexicographic Gröbner base of each prime ideal associated to $\sqrt{\langle S \rangle}$.
- **ExtractTriangular** takes as input a lexicographic Gröbner base and extract the triangular set as described in theorem 3.1.

Algorithm 3

- **Input :** A polynomial system S of equations in $K[X_1, \ldots, X_n]$.
- **Output :** A list of zero-dimensional systems whose roots contain at least one point in each semi-algebraically connected component of $V(S) \bigcap R^n$.

1. list := LexPrimeDecomposition(S), result := [],
2. Choose $A \notin V(S)$.
3. while list $\neq \emptyset$ do
   - S := first(list), and remove S from list, set $d = \mathrm{Dim}(S)$,
   - if $d = 0$ then result := result $\bigcup S$,
   - else
     - $\mathcal{T} = ExtractTriangular(S)$.
     - (*) $Q = \Gamma_A(\mathcal{T}) \bigcup S$ and set $u = \mathrm{Dim}(Q)$
     - if $u = d$ choose another point $A \notin S$ and go to step (*).
     - $d := u$; list := list $\bigcup$ LexPrimeDecomposition(Q),
4. return result.

Remark 3.1 *If $V(\mathcal{G})$ is a variety of dimension d, one needs to compute only d determinants.*

In order to make easier the computations in **LexPrimeDecomposition**, one can reduce the determinants modulo the triangular set at the step (*). Let $\mathrm{prem}(p, q, X)$ denote the classical pseudo-remainder of two polynomials p and q with respect to the variable X. If $p \in K[X_1, \ldots, X_n]$, its reduced form $\mathrm{prem}(p, \mathcal{T})$ can be computed by the following recursive procedure :

- if $\mathcal{T} = \emptyset$, then $\mathrm{prem}(p, \mathcal{T}) = p$.
- else, if X_i is the greatest variable appearing in a polynomial $t \in \mathcal{T}$,

$$\mathrm{prem}(p, \mathcal{T}) = \mathrm{prem}(\mathrm{prem}(p, t, X_i), \mathcal{T} \setminus \{t\}).$$

In particular, there exists polynomials $q_{d+1}, \ldots q_n$ and positive integers $i_{d+1}, \ldots, i_n$ such that :

$$\mathrm{prem}(p, \mathcal{T}) = q_{d+1} t_{d+1} + \ldots + q_n t_n + h_{d+1}^{i_{d+1}} \ldots h_n^{i_n} p.$$

Thus $V(\mathcal{G}) \bigcap V(\mathrm{prem}(p, \mathcal{T})) = V(\mathcal{G}) \bigcap (V(p) \cup V(h_{d+1} \ldots h_n))$. Hence, from theorem 3.1,

$$\dim(V(\mathcal{G}) \bigcap V(p)) < \dim(V(\mathcal{G})) \Longrightarrow \dim(V(\mathcal{G}) \bigcap V(\mathrm{prem}(p, \mathcal{T})) < \dim(V(\mathcal{G})).$$

3.3 Deciding emptiness in any case and computing one point on each semi-algebraically connected component in *generic* cases

Let $\mathcal{G} \subset K[X_1, \ldots, X_n]$ be a lexicographical reduced Gröbner base and $\mathcal{T} = (t_{d+1}, \ldots, t_n)$ a regular separable triangular set extracted from $\mathcal{G}$. Without lost of generality, we suppose that $\forall i \in \{d+1, \ldots, n\}$ $\mathrm{mvar}(t_i) = X_i$.

Definition 3.1 *$\mathcal{T}$ is said to be in quasi-generic position if there exists $k \in \{d+1, \ldots, n\}$ such that*

$$\forall i > k \quad \deg(t_i, X_i) = 1.$$

We denote by k the index of quasi-generic position of $\mathcal{T}_k$.

If $\mathcal{T}$ is in quasi-generic position, we may suppose, without lost of generality, that $t_j = h_j X_j + q_j$ with h_j , $q_j \in K[X_1, \ldots, X_k]$, $\forall j = k+1 \ldots n$.
If $V(\mathcal{G}_k) = \emptyset$ then $V(\mathcal{G}) = \emptyset$.
Suppose that $V(\mathcal{G}_k) \bigcap R^k \neq \emptyset$ and let $M \in \mathcal{D}(V(\mathcal{G}_k, A_k)) \bigcap R^k$ for any $A_k \in R^k$.

- If $M = (x_1, \ldots, x_k) \notin V(h_{k+1})$, there exists a unique value $y \in R$ such that $M' = (x_1, \ldots, x_k, y) \in V(\mathcal{T}_{k+1})$. Moreover, if $M \notin V(\prod_{j=d+1}^{k+1} h_j)$ then, according to theorem 3.1, $M' \in V(\mathcal{G}_{k+1}) \bigcap R^{k+1}$.
- Suppose $M \in V(h_{k+1})$ and $M \notin Sing(V(\mathcal{G}_k))$. Since $dim(V(h_{k+1}) \bigcap V(\mathcal{G}_k)) < dim(V(\mathcal{G}_k))$ and since M is a regular point of $V(\mathcal{G}_k)$, there exists a neighborhood $U \subset V(\mathcal{G}_k) \bigcap R^k$ containing a point N such that $h_{k+1}(N) \neq 0$ and so, according to the preceding item, $N \in V(\mathcal{G}_{k+1}) \bigcap R^{k+1}$.

Recursively, if $(\mathcal{D}(V(\mathcal{G}_k, A_k)) \setminus (Sing(V(\mathcal{G}_k)) \bigcap V(\prod_{i=d+1}^{n} h_i))) \bigcap R^k \neq \emptyset$, then $V(\mathcal{G}) \bigcap R^n \neq \emptyset$.
Since the cases where

$$(\mathcal{D}(V(\mathcal{G}_k), A) \setminus (\mathrm{Sing}(V(\mathcal{G}_k)) \cap V(\prod_{i=d+1}^{n} h_i))) \bigcap R^k = \emptyset$$

$$\text{and } (\mathcal{D}(V(\mathcal{G}_k), A) \bigcap \mathrm{Sing}(V(\mathcal{G}_k)) \bigcap V(\prod_{i=d+1}^{n} h_i)) \bigcap R^k \neq \emptyset,$$

are rare, we propose a specific algorithm, based on **Algorithm 3** and optimized to decide the emptiness.
In the following, we denote by $\Delta(\mathcal{G}_k)$ all the minors of order $k-d$ of the Jacobian matrix associated to $\mathcal{G}_k$ and we define new external functions :

- **ZeroDimTest :** takes as input a zero-dimensional system S and returns *true* if $V(S) \bigcap R^n = \emptyset$, else it returns *false*.
- **ZeroDimSolve :** takes as input a zero-dimensional system S and returns *true* if $V(S) \bigcap R^n =$, else it returns *false*.
- **cleanlingStep :** takes as input a zero-dimensional system S and a polynomial p, and returns a list of real solutions of S which do not vanish p. This can be done by gcd computations on univariate polynomials if we solve zero-dimensional systems by computing Rational Univariate Representations (see [21, 22]).

Algorithm 4

- **Input :** A polynomial system S of equations in $K[X_1, \ldots, X_n]$.
- **Output :** *true* if $V(S) \bigcap R^n = \emptyset$, else it returns *false*.

1. list := LexPrimeDecomposition(S), result := true,
2. Choose $A \notin V(S)$.
3. while list $\neq \emptyset$ do
   - (*)S := first(list), and remove S from list, set $d = \mathrm{Dim}(S)$,
   - if $d = 0$ and if ZeroDimSolve(S) = false then return *false*,
   - $\mathcal{T} = ExtractTriangular(S)$.
   - if $\mathcal{T}$ is in quasi-generic position then
     - newlist := Algorithm3($\Gamma_A(\mathcal{T}_k) \cup S_k$), where k is the index of quasi-generic position of $\mathcal{T}_k$,
     - for S' in newlist, remove S' if ZeroDimSolve(S') = true,
     - if newlist := $\emptyset$ then set result to true and return to step (*),
     - for S' in newlist, remove S' from newlist and if cleaningStep($S', \Delta(\mathcal{G}_k)$) $\neq \emptyset$ then return *false*,
     - for S' in newlist, remove S' from newlist and if cleaningStep($S', \prod_{i=d+1}^{n} h_i$) $\neq \emptyset$ then return *false*,
   - (**) $Q = \Gamma_A(\mathcal{T}) \bigcup S$ and set $u = \mathrm{Dim}(Q)$
   - if $u = d$ choose another point $A \notin S$ and go to step (**).
   - $d := u$; list := list $\bigcup$ LexPrimeDecomposition(Q),
4. return result.

Remark 3.2 *One can guarantee that the method described above computes at least one point in each connected component if* $\forall i \in \{k+1, \ldots, n\}, \quad h_i \in K$, *which occurs, for example, when the system is in Noether position. In other cases, this can not be guaranteed (consider the example :* $t_2 = y - x$ *and* $t_3 = xz - 1$ *with* $z > y > x$ *and take* $A = (0, 1)$*).*

4 Experiments

This section is devoted to present some tests performed with an experimental implementation of our algorithms.

4.1 Software and basic algorithms

The decomposition into primes have been computed using Gb (software devoted to Gröbner bases computations, implemented by J.-C. Faugère) and maple. We first compute a Gröbner base with respect to any ordering (Gb), then deduce, by change of ordering, a lexicographic Gröbner base (Gb) and finally we compute the prime decomposition using multi-variate factorization and gcd's (Maple). In the conclusion of this article, we present some tests made using a recent experimental algorithm (F7 - see [11]) devoted to the decomposition into primes.

The resolution of zero-dimensional systems (counting/isolating of the real roots) has been done using **ZDS** algorithm (Rational Univariate Representation + Isolation of the Real Roots) which uses Gb and RS (software implemented by F. Rouillier). In particular, all the computations have been done using exclusively exact computations.

The other parts of the algorithms were implemented in maple.

In order to show the efficiency of our algorithms, we have applied the CAD algorithm on each example. Remember that this method is more general than ours. In particular, it is currently the only efficient method able to compute, in practice, at least one point on every semi-algebraically connected component of a semi-algebraic variety. The implementation we used (QEPCAD) is build upon the SACLIB library and has been provided by Hoon Hong.

4.2 The methodology

The polynomial systems used for our experiments come from various sources and most of them can be found in the FRISCO Test-Suite (see [13]). A larger list is available on the web page [24].
We may point out that the examples *F633*, *F744* and *F855* come from an industrial application (design of filter banks - see [12]).
All the computations have been performed on a PC Pentium II 400 MHz with 512 Mo of RAM (machine of the UMS Medicis). The timings are given in seconds.
We chose to stop the computations systematically after 12 hours. Also, the symbol ∞ in the timing tables means in fact *stopped after 12 hours.*
It happens that the CAD fails when the number of cells becomes too large. In such cases, we put *failed(n)*, where n denotes a lower bound of the number of cells, in the tables.

4.3 Algorithm 2 / Algorithm 3

The goal of these tests is to show how the use of triangular sets decreases the computation times.

The following table contains the timings for the computation of all the zero-dimensional systems (outputs of Algorithm 2 and Algorithm 3) but excludes the computation times related to their resolution. In the columns *Algorithm 2* and *Algorithm 3* the first number is the cumulative computation time of the prime decompositions, while the second one is the cumulative computation time of the determinants. If one of these both columns contains "?", it means that the preceeding step (either a prime decomposition computation, or a determinant computation) has not ended.

System	Dimension/Degree	Nb Vars	Algorithm 2		Algorithm 3	
Vermeer	1,26	5	0.01	0	0.01	0
Wang	1,114	13	0.12	0	0.12	0
Euler	3,2	10	0.01	0	0.01	0
Neural	1,24	4	0.43	0	0.43	0
Butcher	3,3	8	1.7	0	1.7	0
Buchberger	4,6	8	0	0	0	0
DiscPb	2,3	4	0.02	0	0.02	0
Donati	1,10	4	0.04	26	0.04	0
Hairer2	2,25	13	?	∞	∞	?
Prodecco	2,2	5	284	26	284	0
F633	2,32	10	?	∞	∞	?
F744	1,40	12	24.06	∞	24.06	0.02
F855	1,52	14	5654	∞	5654	173

Table 1 : computation times for Algorithm 2 and Algorithm 3

One can remark that the construction of the zero-dimensional systems is a limiting step in Algorithm 2 and not in Algorithm 3. In Algorithm 3, we compute only a subset of the set of the determinants needed by algorithm 2.

4.4 Algorithm 3 / CAD

4.4.1 Size of the output

In the following table, we give the number of points computed by **Algorithm 3** (sum of the degrees of the zero-dimensional systems) and **QEPCAD** on the examples for which at least one of these methods ends . When **QEPCAD** is stopped after 12 hours, we put ∞ in the table. If the computation failed because the number of cells is too huge, we put failed(n), where n is the lower bound of number of cells that **QEPCAD** has predicted.

System	Algorithm 3 + ZDS	QEPCAD
Vermeer	84	65976
Wang	132	∞
Euler	10	failed(872043)
Neural	133	205
Butcher	15	∞
Buchberger	32	failed(991324)
DiscPb	28	∞
Donati	61	10

Table 2 : comparison between (Algorithm 3 + ZDS) and QEPCAD

One can observe that these results are coherent with the theoretical complexity : the output of the CAD is doubly exponential in the number of variables while the number of points of the output of our algorithm is proportional to the number of semi-algebraically connected components of the real algebraic variety and thus singly exponential in the number of variables. We can also remark that none of the methods solved the examples *Hairer2*, *Prodecco*, *F633*, *F744* and *F855*, even if *Algorithm 3* provided all the zero-dimensional systems. These systems were too large for the computation of a Gröbner base by Gb.

4.4.2 Computation times

One of our motivations was to provide an algorithm whose output is reasonable with the hope to get significantly better computation times, compared to existing implementations that computes at least the same thing, even if the methods used have not, theoretically, a better complexity in terms of computation times.
The next table shows that both algorithms **Algorithm 2 + ZDS** and **Algorithm 3 + ZDS** have a better behavior, in practice, than **QEPCAD** :

System	Algorithm 2 + ZDS	Algorithm 3 + ZDS	QEPCAD
Vermeer	62.36	3.32	43
Wang	1.37	1.37	∞
Euler	0.01	0.01	failed(*872043*)
Neural	1.02	1.02	0.9
Butcher	1.7	1.7	∞
Buchberger	< 0.01	< 0.01	failed(*991324*)
DiscPb	0.2	0.2	∞
Donati	11609	10	0.6

Table 3 : Computation times for Algorithm 2 +ZDS, Algorithm 3 +ZDS and QEPCAD.

4.5 Algorithm 4

The last table shows the progress induced by **Algorithm 4**. According to remark 3.2, **Algorithm 4** computes one point on each connected component in favorable cases, and allows to decide if a variety is empty or not in any case. The examples for which **Algorithm 4** gives at least one point on each semi-algebraically connected component are marked by *.

System	Algorithm 2 + ZDS	Algorithm 3 + ZDS	Algorithm 4	QEPCAD
Vermeer	62.36	3.32	<0.01	43
Wang	1.37	1.37	0.13	∞
Euler	0.01	0.01	<0.01*	failed(*872043*)
Neural	1.02	1.02	0.44*	0.9
Butcher	1.7	1.7	1.7*	∞
Buchberger	< 0.01	<0.01	<0.01*	failed(*991324*)
DiscPb	0.2	0.2	0.02	∞
Donati	11609	10	0.04	0.6
Hairer2	∞	∞	23.03	failed(*872043*)
Prodecco	∞	∞	286	∞
F633	∞	∞	5700	∞
F744	∞	∞	40	∞
F855	∞	∞	5664	∞

Table 4 : Outputs of Algorithm 2 + ZDS, Algorithm 3 + ZDS and QEPCAD.

In terms of computations, the difference between **Algorithm 3** and **Algorithm 4** is the number and size of the intermediate determinants. One can see that the zero-dimensional systems provided by **Algorithm 4** are much more simple to solve.

The cases where **Algorithm 4** computes one point on each semi-algebraic component are few which means, in particular, that, in our test list, the systems in Noether position are few and so justify a large part of our study whose objective is to provide an algorithm that works in every situation.

5 Conclusions

We have provided an efficient algorithm (Algorithm 3) that allows to compute one point on each semi-algebraically connected component of a real algebraic variety, without assumption neither on the variety (smoothness, compactness) nor on the system of polynomial equations that define it.

We proposed an optimization (Algorithm 4) for deciding the emptiness of the variety in any cases or for computing at least one point on each connected component in *generic* cases (see remark 3.2). According to the experiments, we noticed that in practice, these conditions of genericity (for example the Noether position) are too strong, which prevents **algorithm 4** for computing at least one point on each semi-algebraically connected component.

Moreover, we will have much better timings in a near future. For example, we try a recent prototype, due to J.C. Faugère, of an algorithm for computing prime decompositions that speeds up our algorithms : with this implementation, *Algorithm 4* can solve *F633* in 7.2 sec. and *F855* in 26 sec.

According to other experiments we made, additional assumptions on the variety (smoothness, compactness) or on the system of equations that defines it (Noether position, radical, prime, etc ...) speeds up strongly the method. For example, if we suppose the real algebraic set to be compact,

then, according to Lemma 3.1 we can replace the distance function by any projection with respect to one coordinate X_i. In practice, it is sufficient to replace $\overrightarrow{AM}$ by the vector $\overrightarrow{u_i}$ whose coordinates are null except the i-th.

The theoretical complexity of our method depends strongly on the complexity of the prime decomposition. So note that there is no precise result about it. We can just give an upper doubly exponential in the number of variables bound (since we compute lexicographical Gröbner bases).

We plan to extend our work to the case of semi-algebraic sets, generalizing our main results or simply applying well known transformations (see for example [25]) that comes to study real algebraic varieties.

References

[1] P. Aubry, *Ensembles triangulaires de polynômes et résolution de systèmes algébriques. Implantation en Axiom*, Doctoral Thesis, University of Paris VI, 1999.

[2] P. Aubry, D. Lazard, M. Moreno Maza, *On the theories of triangular sets,* in Journal of Symbolic Computation, 1999.

[3] B. Bank, M. Giusti, J. Heintz, and M. MBAKOP *Polar Varieties and Efficient Real Elimination*, to appear in MSRI journal, (2000).

[4] S. Basu, R. Pollack, M.-F. Roy, *A New Algorithm to Find a Point in Every Cell Defined by a Family of Polynomials,* in Quantifier Elimination and Cylindrical Algebraic Decomposition, Texts and Monographs in Symbolic Computation, B. Caviness and J. Johnson, Eds. 341-349, Springer-Verlag, Wien, New York (1998).

[5] S. Basu, R. Pollack, M.-F. Roy, *On the combinatorial and algebraic complexity of Quantifier elimination.* J. Assoc. Comput. Machin., 43, 1002–1045, (1996).

[6] J. Bochnak, M. Coste, M.-F. Roy, *Real algebraic geometry,* Springer-Verlag (1999).

[7] J. Canny , *A toolkit for nonlinear algebra,* Goldberg, Ken (ed.) et al., Algorithmic foundations of robotics, Proceedings of the workshop on the algorithmic foundations of robotics, WAFR '94, held in San Francisco, CA, USA, 17-19 February, 1994. Wellesley, MA: A.K. Peters.

[8] G. E. Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decomposition,* Springer Lecture Notes in Computer Science 33, 515- 532, (1975).

[9] P. Conti, C. Traverso, *Algorithms for the real radical,* unpublished manuscript

[10] D. Cox, J. Little, D. O'Shea, *Ideals, Varieties, and Algorithms,* Springer-Verlag (1991)

[11] J.C. Faugère, *FGb*, available on the web http://www-calfor.lip6.fr/ jcf

[12] , J.C. Faugère and F. Moreau de Saint Martin and F .Rouillier *Design of regular nonseparable bidimensional wavelets using Groebner basis techniques*, in IEEE SP Transactions Special Issue on Theory and Applications of Filter Banks and Wavelets (1997).

[13] The FRISCO Test-suite, available on the web http://www-sop.inria.fr/saga/POL

[14] D. Grigor'ev, N. Vorobjov , *Solving Systems of Polynomial Inequalities in Subexponential Time,* J. Symbolic Comput., 5:37–64, (1988).

[15] J. Heintz, M.-F. Roy, P. Solernó , *On the Complexity of Semi-Algebraic Sets,* Proc. IFIP 89, San Francisco. North-Holland 293-298 (1989).

[16] M. Kalkbrenner, *Three contributions to elimination theory,* Doctoral thesis, 1991.

[17] D. Lazard, *A new method for solving algebraic systems of positiove dimension,* in Discrete Applied Mathematics, 1991.

[18] M. Moreno Maza, *Calculs de Pgcd au-dessus des Tours d'Extensions Simples et Résolution des Systèmes d'Equations Algébriques,* Doctoral Thesis, University of Paris VI, 1997.

[19] D. Mumford *Algebraic Geometry I, Complex projective varieties*, Berlin, Heildelberg, New York : Springer Verlag (1976).

[20] J. Renegar *On the computational complexity and geometry of the first order theory of the reals,* J. of Symbolic Comput.13(3):255-352, (1992).

[21] F. Rouillier, *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux,* Doctoral Thesis, University of Rennes I (1996).

[22] F. Rouillier, *Solving Zero-Dimensional Systems through the Rational Univariate Representation,* AAECC Journal.9 : 433-461 (1999).

[23] F. Rouillier, M.-F. Roy, M. Safey El Din, *Finding at least one point in each connected component of a real algebraic set defined by a single equation,* to appear in Journal of Complexity, 1999.

[24] F. Rouillier, M. Safey El Din, *Some Benchmarks for RSDF Algorithm,* available on the Web http://posso.lip6.fr/ safey/benchs.html

[25] M.-F. Roy, *Basic algorithms in real algebraic geometry: from Sturm theorem to the existential theory of reals,* Lectures on Real Geometry in memoriam of Mario Raimondo, Expositions in Mathematics 23, 1- 67. Berlin, New York: de Gruyter (1996).

[26] A. Seidenberg, *A new decision method for elementary algebra,* Annals of Mathematics, 60:365–374, (1954).