Efficient Polynomial Equation Solving: Algorithms and Complexity

Juan Sabia*

Abstract. These notes intend to familiarize the reader with some algorithmic notions concerning polynomial equation system solving. We start by dealing with the dense representation of multivariate polynomials. Some results about the algebraic complexities of the effective Nullstellensatz, of quantifier elimination processes and of the decomposition of varieties when using this model are mentioned. Then, it is shown that these complexities are essentially optimal in the dense representation model. This introduces a change in the encoding of polynomials so as to get better upper bounds for the complexities of the mentioned algorithms: the straight-line program representation is defined and discussed. Some complexity results in the straight-line program representation model are shown (effective Nullstellensatz, quantifier elimination procedures, for instance). A final comment on the Newton-Hensel method to approximate roots is made.

1 Introduction

The fundamental problem we are going to deal with is to solve a system of multivariate polynomial equations with coefficients in $\mathbb{Q}$. Of course, we should define what to solve such a system is. A possible first approach would be to decide whether there are solutions to the system, and if there are solutions, to describe them in a 'useful' (or at least in an easier) way.
Many attempts to do this are based on trying to transform our problem into a Linear Algebra one. The reason of this is that we know how to solve many Linear Algebra problems.
The focus of our attention will be the *algorithmic* solutions of these problems and we are going to classify somehow the algorithms we use. Roughly speaking, the less time an algorithm takes to perform a task, the better. This will lead to the definition of *algebraic complexity*, a kind of measure for the time an algorithm takes.
One of the problems we have when we deal with multivariate polynomials is that the known effective ways to factorize them take a lot of time, so we will try not to use this tool within our algorithms.
In the different sections, we are going to state the problems that will be taken into account and describe (or just mention, if the description is beyond the scope of these notes) some ways of solving them.

Before we begin considering the problems we need to fix some notation and give some definitions:

A typical system of polynomial equations is

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \ldots \\ f_s(x_1, \ldots, x_n) = 0 \end{cases}$$

*Depto. de Matemática - Fac. de Cs. Exactas y Naturales - Univ. de Buenos Aires and CONICET - Argentina

where $f_1, \ldots, f_n$ are polynomials in $\mathbb{C}[X_1, \ldots, X_n]$. Whenever we want to speak about a group of variables, we may use just a capital letter with no index; for example in this case, we could have written $\mathbb{C}[X]$. The set of solutions $V \subset \mathbb{C}^n$ of such a system will be called an *algebraic variety*. Its *dimension* has to do with the minimun number of 'generic' hyperplanes we have to cut V with so as to get the empty set. For example, a point has dimension zero (a generic hyperplane does not cut it); a line has dimension one (a generic hyperplane cuts it, but two generic hyperplanes do not), etc. For a more precise definition of dimension see for example [22] or [4].
From the algorithmic point of view, we will deal strictly with polynomials over $\mathbb{Q}$ but we still want to consider all their solutions in $\mathbb{C}^n$.

Sometimes we will be dealing with other fields than $\mathbb{Q}$ and $\mathbb{C}$. If k is a field, $\overline{k}$ will denote an algebraic closure of k.

2 Statement of the Problems

In this section we are going to state some of the problems we usually deal with when we try to solve a system of polynomial equations.

2.1 Effective Hilbert's Nullstellensatz

Let $X_1, \ldots, X_n$ be indeterminates over $\mathbb{Q}$. Given s polynomials $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$ if we want to solve the system of polynomials equations

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \ldots \\ f_s(x_1, \ldots, x_n) = 0 \end{cases}$$

the very first question we would like to answer is whether there exists any point $(x_1, \ldots, x_n) \in \mathbb{C}^n$ satisfying this system (that is to say, if the equations $f_1 = 0, \ldots, f_s = 0$ share a common solution in $\mathbb{C}^n$).

If the polynomials $f_1, \ldots, f_s$ have all degrees equal to 1, the system we are dealing with is a linear system and there is a simple computation of ranks involving the coefficients of the polynomials which answers our question:
Suppose our linear system is given by $A.X^t = B$ (with $A \in \mathbb{Q}^{s \times n}$ and $B \in \mathbb{Q}^{s \times 1}$). Then

$$\exists x \in \mathbb{C}^n \ / \ A.x^t = B \iff \operatorname{rank}(A) = \operatorname{rank}(A|B).$$

The first step towards a generalization of this result when we deal with polynomials of any degree is the following well-known theorem.

Theorem 2.1 *(Hilbert's Nullstellensatz) Let $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$. Then the following statements are equivalent:*

i) $\{x \in \mathbb{C}^n \ / \ f_1(x) = \ldots = f_n(x) = 0\} = \emptyset$.

ii) There exist polynomials $g_1, \ldots, g_s \in \mathbb{Q}[X_1, \ldots, X_n]$ such that $\sum_{1 \leq i \leq s} g_i.f_i = 1$.

A proof of this Theorem can be found in almost any basic textbook on Algebraic Geometry (see for example [12] or [4]). This result was already known by Kronecker and it essentially shows how a geometric problem (Is the variety defined as the common zeroes of a fixed set of polynomials empty?) is equivalent to an algebraic one (Is 1 in the ideal $(f_1, \ldots, f_s)$?).

We will call *Effective Hilbert's Nullstellensatz* to any algorithm that, given as input the polynomials $f_1, \ldots, f_s$, provides us with polynomials $g_1, \ldots, g_s$ (if they exist) such that $\sum_{1 \leq i \leq s} g_i . f_i = 1$. Later on, we will mention some Effective Hilbert's Nullstellensätze.

2.2 Effective Equidimensional Decomposition

Supposing we already know that a particular system of polynomial equations has solutions, we may need to answer some questions about the geometry of the set of solutions in $\mathbb{C}^n$: Does it consist only of finitely many points? Is there a whole curve of solutions? Are there isolated solutions?, etc.
All these questions can be answered by means of geometric decompositions of the algebraic variety defined by the system of polynomials.

Definition 2.2 *An algebraic variety $C \subset \mathbb{C}^n$ is* **irreducible** *if it satisfies*

$$C = C_1 \cup C_2 \text{ where } C_1 \text{ and } C_2 \text{ are algebraic varieties} \Rightarrow C = C_1 \text{ or } C = C_2.$$

The following is a classical result from Algebraic Geometry. Its proof can be found, for example, in [22] or [4].

Proposition 2.3 *(Irreducible decomposition) Let $V \subset \mathbb{C}^n$ be an algebraic variety. Then, there exist unique irreducible varieties $C_1, \ldots, C_r$ such that $C_i \not\subset C_j$ if $i \neq j$ and*

$$V = \bigcup_{1 \leq i \leq r} C_i.$$

From our point of view and our definitions, the irreducible decomposition is not algorithmically attainable. Roughly speaking, if this were so, just by considering the case $n = 1$, we would be able to find all the roots of any one-variate rational polynomial (note that the irreducible decomposition of $\{x \in \mathbb{C} \ / \ \prod_{1 \leq i \leq d}(x - \alpha_i) = 0\}$ is exactly $\bigcup_{1 \leq i \leq d}\{\alpha_i\}$). This is why we are going to consider a less refined decomposition of a variety.

Let $V = \bigcup_{1 \leq i \leq r} C_i$ be the irreducible decomposition of the variety V and, for every $0 \leq j \leq n$, consider the union of all the irreducible components of V of dimension j

$$V_j := \bigcup_{\{i \ / \ 1 \leq i \leq r \text{ and } \dim C_i = j\}} C_i.$$

It is obvious that $V = \bigcup_{0 \leq j \leq n} V_j$ and this unique decomposition is called the **irredundant equidimensional decomposition** (or equidimensional decomposition for short) of V.

Note that the information yielded by this decomposition still allows us to answer the questions we asked above. For example, a non-empty variety V consists only of finite points if and only if $V_0 \neq \emptyset$ and $V_i = \emptyset$ for every $1 \leq i \leq n$.

The equidimensional decomposition has the following property, nice from the algorithmic point of view:

Proposition 2.4 *Let $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$ be polynomials and let $V \in \mathbb{C}^n$ be the algebraic variety of their common zeroes. If V_j is one of the components appearing in the irredundant equidimensional decomposition, then there exist polynomials $g_1, \ldots, g_\ell \in \mathbb{Q}[X_1, \ldots, X_n]$ defining V_j.*

The core of this result is that there are **rational** polynomials defining V_j, so we have a chance to compute the irredundant equidimensional decomposition algorithmically using only rational coefficients.
Therefore, we call *effective equidimensional decomposition* to any algorithm that, given an algebraic variety $V \subset \mathbb{C}^n$ defined by rational polynomials, describes its equidimensional components as separate varieties.

2.3 Effective Quantifier Elimination

A well-known result from Logic states that any first order formula with coefficients in an algebraically closed field is equivalent to another formula without quantifiers (see [2] for details).
For the sake of simplicity, we will state this result in a very particular case:

Theorem 2.5 *Let $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ be indeterminates over $\mathbb{Q}$ and let $f_1, \ldots, f_s, g_1, \ldots, g_t \in \mathbb{Q}[X_1, \ldots, X_n, Y_1, \ldots, Y_m]$ be polynomials. Let*

$$V := \{x \in \mathbb{C}^n \ / \ \exists y \in \mathbb{C}^m \ : \ f_1(x,y) = 0 \wedge \ldots \wedge f_s(x,y) = 0 \wedge g_1(x,y) \neq 0 \wedge \ldots \wedge g_t(x,y) \neq 0\}.$$

Then, there exists a **quantifier free** *formula φ involving polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$, equalities, inequalities and the symbols $\wedge$ and $\vee$ such that*

$$V = \{x \in \mathbb{C}^n \ / \ \varphi(x)\}.$$

Let's give some examples to make this clearer.

Example 1: Suppose we are given a generic polynomial of degree bounded by d in one variable and we want to know if it has a root in $\mathbb{C}$. Then, the set we want to describe is

$$V := \{(x_0, x_1, \ldots, x_d) \in \mathbb{C}^{d+1} \ / \ \exists y \in \mathbb{C} \ : \ x_d y^d + x_{d-1} y^{d-1} + \ldots + x_0 = 0\}.$$

Evidently, the Fundamental Theorem of Algebra states that a quantifier-free way of defining V is

$$V = \{(x_0, x_1, \ldots, x_d) \in \mathbb{C}^{d+1} \ / \ x_0 = 0 \vee x_1 \neq 0 \vee x_2 \neq 0 \vee \ldots \vee x_d \neq 0\}.$$

Example 2: A very well-known example of quantifier elimination procedure from Linear Algebra is the use of the determinant. The set

$$V := \{(x_{ij}) \in \mathbb{C}^{n \times n} \ / \ \exists (y_1, \ldots, y_n, y'_1, \ldots, y'_n) \in \mathbb{C}^{2n} \ : \ (y_1, \ldots, y_n) \neq (y'_1, \ldots, y'_n) \wedge$$

$$\wedge \left\{ \begin{array}{l} x_{11} y_1 + \ldots + x_{1n} y_n = x_{11} y'_1 + \ldots + x_{1n} y'_n \\ \ldots \\ x_{n1} y_1 + \ldots + x_{nn} x_n = x_{n1} y'_1 + \ldots + x_{nn} y'_n \end{array} \right\}$$

is exactly the subset of $\mathbb{C}^{n \times n}$

$$V = \{(x_{ij}) \in \mathbb{C}^{n \times n} \ / \ \det(x_{ij}) = 0\}.$$

Example 3: The resultant with respect to a single variable Y between two generic polynomials $f_1, f_2 \in \mathbb{Q}[X_1, \ldots, X_n][Y]$ monic in Y and of degree r and s respectively is another example of eliminating polynomial:

$$\{x \in \mathbb{C} \ / \ \exists y \in \mathbb{C} \ : \ f_1(x,y) = 0 \wedge f_2(x,y) = 0\} = \{x \in \mathbb{C} \ / \ \mathrm{Res}_Y(f_1(x,Y), f_2(x,Y)) = 0\}.$$

As before, we will say that we have an *efficient quantifier elimination procedure* if we have an algorithm that, from a formula

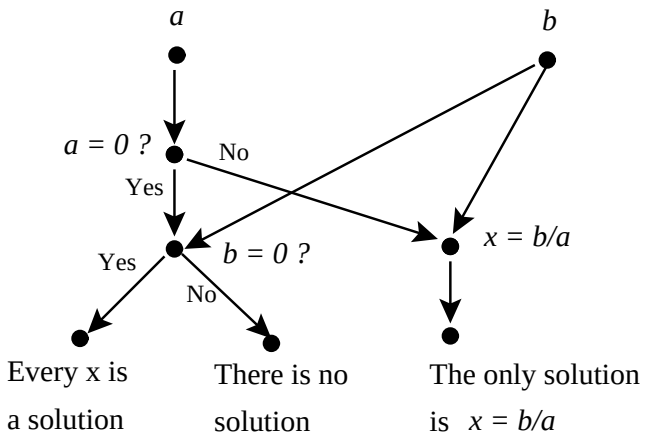
$$\exists y \in \mathbb{C}^m \ : \ f_1(x_1, \ldots, x_n, y) = 0 \ \wedge \ldots \wedge \ f_s(x_1, \ldots, x_n, y) = 0 \wedge$$

$$\wedge \ g_1(x_1, \ldots, x_n, y) \neq 0 \ \wedge \ldots \wedge \ g_t(x_1, \ldots, x_n, y) \neq 0,$$

produces a quantifier-free formula φ defining the same subset of $\mathbb{C}^n$.

3 Algorithms and Complexity

We have defined efficiency based on the existence of algorithms performing different tasks. But what is an algorithm? It is time we defined it.
The idea of algorithm we deal with is the following: given some data in a certain way (numbers, formulae, etc), an algorithm will be a sequential list of operations or comparisons that ends in some logical or mathematical 'object' we would like to compute. For example, suppose you want an algorithm to solve the equation $ax + b = 0$ with coefficients in $\mathbb{Q}$ and that you can deal with rational numbers algorithmically. A possible algorithm to do this would be



Speaking a little more formally, ours algorithms would be *directed acyclic graphs*. Each node of a graph would represent an element of $\mathbb{Q}$, an operation or a comparison between two elements of $\mathbb{Q}$. Each 'incoming' arrow denotes that the previously computed element or condition is needed to perform the following operation.

The idea of complexity has to do with measuring how long it would take to perform the algorithm. The more complicated our graph is, the longer our algorithm will take. So, a first measure of complexity to be taken into account is the number of nodes of the graph and this is the notion of **complexity** we are going to use throughout these notes.
Needless to say, this measure of complexity is not accurate at all. For example, to perform the sum $1 + 1$ is much more simpler for a machine than to add huge numbers. Moreover, a sum is generally simpler that a product. These considerations give place to a number of different kinds of complexities (non-scalar complexity, bit complexity, etc) which we will not take into account. But, of course, if an algorithm has a very large complexity in our terms, then it will be useless to try to run it on any computer.
There are other possible variables to be taken into account when considering an algorithm: the space in memory needed to perform it or whether it can be successfully parallelized or not, for example, but we are not going to consider these aspects in these notes.
We say an algorithm is **polynomial** when its complexity is bounded by a polynomial function in the size of the data given to perform the algorithm (called the *input*).
We will also use the usual $\mathcal{O}$ notation: given two functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$, we say that $f = \mathcal{O}(g)$ if and only if there exists $k \in \mathbb{N}$ such that $f(x) \leq kg(x)\ \forall x \in \mathbb{N}$.

4 Dense Encoding and Algorithms

Evidently, as we are trying to solve algorithmic problems involving polynomials, we need to encode them. The first (and most naive) way of encoding a polynomial is to write it as the vector of its coefficients. To do this, we need to know a bound for the degree of the polynomial and the number of variables involved in advance.
For example, let $f(X, Y) = X^2 - 2XY + Y^2 + 3$ be a polynomial we want to encode. As we know $\deg(f) = 2$, we only have to store the coefficients of the polynomial up to this degree. We previously fix an order of all the monomials up to degree 2, for example $(1, X, X^2, XY, Y, Y^2)$ and, using this order, the polynomial f will be encoded as $(3, 0, 1, -2, 0, 1)$.
This way of encoding polynomials is called *dense form*.
Let's consider the possible complexity of a polynomial f of degree bounded by d in n variables; that is to say, how many numbers (nodes of a graph) will be needed to encode it. According to our definition, we have to compute how many monomials in n variables of degree bounded by d there are, and the exact number is $\binom{d+n}{d}$. If we consider that we are working with a fixed number of variables $n \geq 2$ but that the degrees can change, we have that

$$\binom{d+n}{d} = \prod_{1 \leq i \leq n} \frac{d+i}{i} \leq d^n.$$

Furthermore, asymptotically in d we have that these two quantities are of the same order because

$$\frac{d^n}{\prod_{1 \leq i \leq n} \frac{d+i}{i}} \leq n!$$

and that is why we consider a polynomial of degree d in n variables has $\mathcal{O}(d^n)$ coefficients.

Another way of encoding special multivariate polynomials is the *sparse form*, which consists in just specifying which monomials take non-zero coefficients and which are these coefficients, but we are not going to use this encoding in these notes.

4.1 Hilbert's Nullstellensatz

As we have seen in Section 2.1, an effective Hilbert's Nullstellensatz is any algorithm that, given as input the polynomials $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$, decides whether there exists polynomials $g_1, \ldots, g_s \in \mathbb{Q}[X_1, \ldots, X_n]$ such that

$$\sum_{1 \leq i \leq s} g_i . f_i = 1 \tag{1}$$

and computes a particular solution $(g_1, \ldots, g_s)$ to this identity.
A major goal would be to bound the possible degrees of some polynomial solutions $g_1, \ldots, g_s$ to equation (1) as a function of s, n and a bound d for the degrees of the polynomials $f_1, \ldots, f_s$. If we were able to do so, our problem could be easily transformed into a Linear Algebra problem: we can write new variables for the coefficients of the polynomials $g_1, \ldots, g_s$ and equation (1) turns into a linear system by identifying the coefficients on the left with those on the right. That is why, sometimes, an effective Hilbert's Nullstellensatz can be thought as the following problem:

Show explicitly a function $\varphi : \mathbb{N}^3 \to \mathbb{N}$ satisfying the following property:
Let $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$ such that $\deg(f_i) \leq d$ $(1 \leq i \leq s)$. If $1 \in (f_1, \ldots, f_s)$, there exist polynomials $g_1, \ldots, g_s \in \mathbb{Q}[X_1, \ldots, X_n]$ with $\deg(g_i) \leq \varphi(n, s, d)$ $(1 \leq i \leq s)$ such that $\sum_{1 \leq i \leq s} g_i . f_i = 1$.

Just as an example, we are going to show a very elementary theorem of this kind.

Theorem 4.1 *Let $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$ such that $\deg(f_i) \leq d$ $(1 \leq i \leq s)$. If $1 \in (f_1, \ldots, f_s)$, there exist polynomials $g_1, \ldots, g_s \in \mathbb{Q}[X_1, \ldots, X_n]$ with $\deg(g_i) \leq (2d)^{2^n}$ $(1 \leq i \leq s)$ such that $\sum_{1 \leq i \leq s} g_i . f_i = 1$.*

Proof.
We shall prove this using induction on n. For $n = 1$, the result is true for $\deg g_i \leq d - 1$ (using the Euclid algorithm properly).
Suppose the result is true for n. Let now $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_{n+1}]$ such that $\deg(f_i) \leq \deg(f_1) = d$ $(2 \leq i \leq s)$.
Making a linear change of variables if necessary, we can suppose every polynomial f_i is monic in X_1. Introduce new variables $U_1, \ldots, U_s, V_1, \ldots, V_s$ and consider the polynomials

$$F := \sum_{1 \leq i \leq s} U_i f_i \quad \text{and} \quad G := \sum_{1 \leq i \leq s} V_i f_i \quad \text{in} \quad \mathbb{Q}[U_1, \ldots, U_s, V_1, \ldots, V_s][X_1, \ldots, X_{n+1}].$$

Consider the resultant

$$\mathrm{Res}_{X_1}(F, G) \in \mathbb{Q}[U_1, \ldots, U_s, V_1, \ldots, V_s][X_2, \ldots, X_{n+1}].$$

Note that this resultant is bi-homogeneous in the groups of variables (U, V) of bi-degree (d, d). If we write

$$\mathrm{Res}_{X_1}(F, G) = \sum_{\alpha, \beta} h_{\alpha,\beta}(X_2, \ldots, X_{n+1}) U^\alpha V^\beta,$$

we are going to prove that $f_1, \ldots, f_s$ have a common root in $\mathbb{C}^{n+1}$ if and only if $(h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d}$ have a common root in $\mathbb{C}^n$.
If $(x_1, \ldots, x_{n+1}) \in \mathbb{C}^{n+1}$ is a common root of $f_1, \ldots, f_s$, then $\mathrm{Res}_{X_1}(F, G)(x_2, \ldots, x_{n+1})(U, V) = 0$ and therefore, $(h_{\alpha,\beta})_{|\alpha|=d, |\beta|=d}$ have a common root in $\mathbb{C}^n$.

On the other hand, if $(x_2, \ldots, x_{n+1})$ is a common root of the polynomials $(h_{\alpha,\beta})_{|\alpha|=d,|\beta|=d}$, consider the polynomials F and G in $\mathbb{Q}(U_1, \ldots, U_s, V_1, \ldots, V_s)[X_1, x_2 \ldots, x_{n+1}]$. Then, $F(X_1, x_2, \ldots, x_{n+1})$ and $G(X_1, x_2, \ldots, x_{n+1})$ share a common root in $\overline{\mathbb{Q}(U, V)}$. But, as the roots of $F(X_1, x_2, \ldots, x_{n+1})$ lie in $\overline{\mathbb{Q}(U)}$ and the roots of $G(X_1, x_2, \ldots, x_{n+1})$ lie in $\overline{\mathbb{Q}(V)}$, the common root must be in $\mathbb{C}$. That is, there exists $x_1 \in \mathbb{C}$ such that $F(x_1, \ldots, x_{n+1}) = 0$ and $G(x_1, \ldots, x_{n+1}) = 0$. As the variables U, V are algebraically independent, we conclude that $(x_1, \ldots, x_{n+1})$ is a common root of the polynomials $f_1, \ldots, f_s$.
Then we have reduced the number of variables in one. Note that, because of Hilbert Nullstellensatz, we have shown that

$$1 \in (f_1, \ldots, f_s) \iff 1 \in (h_{\alpha,\beta})_{|\alpha|=d,|\beta|=d}.$$

$\mathrm{Res}_{X_1}(F, G)$ can be written as a linear combination of F and G. Taking into account the degrees of the polynomials involved, we can state that there exists polynomials R and S in $\mathbb{Q}[U, V][X]$ of degree bounded by $2d^2$ in the variables X such that $\mathrm{Res}_{X_1}(F, G) = RF + SG$. Just rewriting this identity into powers of U and V, we have that

$$h_{\alpha,\beta} = \sum_{1 \le i \le s} p_i^{(\alpha,\beta)} f_i$$

where the polynomials $p_i^{(\alpha,\beta)}$ have degrees bounded by $2d^2$. Using inductive hypothesis for the polynomials $\{h_{\alpha,\beta}\}_{(|\alpha|=d,|\beta|=d)}$ whose degrees are bounded by $2d^2$, the theorem follows. □

Evidently, this kind of bound is not good for algorithmic purposes. There are much better bounds for the degrees of the polynomials appearing in the Nullstellensatz but the proofs are beyond the scope of this course. We can mention among them [5], [15], [20] and [23].

Let's make a final comment on the complexity of an algorithm that, using the dense encoding of polynomials, decides whether the variety they define is empty or not and, if it is empty, gives as output a linear combination of the input polynomials equal to 1.
If the input polynomials $f_1, \ldots, f_s$ have degree bounded by d and the degrees of the polynomials involved in the linear combination given by the Nullstellensatz is $\varphi(d, n, s)$, then we only need to solve a system of $\mathcal{O}(\varphi(d, n, s) + d)^n$ linear equations in $\mathcal{O}(s\varphi(d, n, s)^n)$ variables (or to prove that this system has no solution). The complexity of doing this is of order $\mathcal{O}(s^4.(\varphi(d, n, s) + d)^{4n})$.
Therefore, using the best Effective Nullstellensäzte known up to now, that essentially states $\varphi(d, n, s) = d^n$, the bounds of an algorithm using dense encoding will be at least of order $\mathcal{O}(sd^{n^2})$.

4.2 Quantifier Elimination

Suppose now we are given $s + t$ polynomials in $\mathbb{Q}[X_1, \ldots, X_n][Y_1, \ldots, Y_m]$ of degrees bounded by d and we want to give algorithmically a quantifier-free formula equivalent to

$$\exists y \in \mathbb{C}^m \ : \ f_1(x, y) = 0 \wedge \ldots \wedge f_s(x, y) = 0 \wedge g_1(x, y) \neq 0 \wedge \ldots \wedge g_t(x, y) \neq 0. \tag{2}$$

Rabinowicz's trick allows us to consider only equalities by means of a new indeterminate Z and therefore, the previous formula is equivalent to

$$\exists y \in \mathbb{C}^m \ z \in \mathbb{C} \ : \ f_1(x, y) = 0 \wedge \ldots \wedge f_s(x, y) = 0 \wedge \big(1 - z. \prod_{1 \le i \le t} g_i(x, y)\big) = 0.$$

Fixing $x \in \mathbb{C}^n$ and using Hilbert's Nullstellensatz, this last formula is equivalent to

$$\nexists p_1, \ldots, p_s, p_{s+1} \in \mathbb{C}[Y_1, \ldots, Y_m, Z] \ / \ 1 = \sum_{1 \leq i \leq s} p_i f_i + p_{s+1}\big(1 - Z. \prod_{1 \leq i \leq t} g_i\big).$$

Any effective Hilbert's Nullstellensatz providing upper bounds for the degrees of the polynomials p_i involved allows us to translate this last problem into a Linear Algebra problem in the coefficients of the polynomials f_i and g_j. Suppose the linear system involved is $A.X^t = B$ where $A \in \mathbb{C}^{\ell \times k}$ and $B \in \mathbb{C}^{\ell}$. The non-existence of solutions is equivalent to the condition $\text{rank}A \neq \text{rank}(A|B)$. Using that the rank of a matrix can be computed by means of the determinants of its minors, this last condition can be translated into a (very long) formula involving $\wedge$, $\vee$, equalities and inequalities to zero. This formula works for every $x \in \mathbb{C}^n$ and therefore, this formula is equivalent to formula (2).
It is evident that the better the effective Nullstellensatz we are using, the smaller the complexity of this kind of algorithm will be.

4.3 Equidimensional Decomposition

There are many algorithms computing equidimensional decomposition. In the case of dense encoding, we can mention the algorithm by Chistov and Grigor'ev (see [3]) and the one by Giusti and Heintz (see [9]). Although the proof of this last result is not to be included here, we can state their main theorem and the complexity obtained:

Theorem 4.2 *Let $f_1, \ldots, f_s$ polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$ of degree bounded by d and let V be the variety they define. There exists an algorithm of complexity $s^5 d^{O(n^2)}$ which computes, for every $0 \leq i \leq n$, $d^{O(n^2)}$ polynomials of degree bounded by d^n defining the equidimensional component of V of dimension i.*

4.4 Life is Not Easy

In this section, we are going to show that the better bounds already obtained (and mentioned) for the efficient Hilbert's Nullstellensatz are of the best possible order.

There is a very well-known example by Masser and Philippon that gives a very high lower bound for the degrees in the Nullstellensatz:
Take

$$f_1 = X_1^d, f_2 = X_1 - X_2^d, \ldots, f_{n-1} = X_{n-2} - X_{n-1}^d, f_n = 1 - X_{n-1} X_n^{d-1}.$$

If $g_1, \ldots, g_n$ are polynomials such that $1 = \sum_{1 \leq i \leq n} g_i f_i$, then, specializing the variables in suitable expressions in a new indeterminate T such that all the polynomials f_i vanish for $2 \leq i \leq n$, we have that

$$1 = g_1(T^{(d-1)d^{n-2}}, \ldots, T^{d-1}, \frac{1}{T})T^{(d-1)d^{n-1}}$$

and therefore, $\deg_{X_n} g_1 \geq (d-1)d^{n-1}$.
That is to say, the complexities of any algorithm using effective Linear Algebra in the way we stated before must be greater or equal to $\mathcal{O}(d^{n^2})$.

5 Straight-line Program Encoding for Polynomials

The previous observation shows that it is impossible to obtain more efficient general algorithms when dealing with dense encoding of polynomials. There are at least two ways of avoiding this problem: the first one is to change the way the polynomials are encoded (that is to say, to try to find a shorter encoding for polynomials); the second one is to design non-general algorithms which can only solve special problems but within a lower complexity. We are going to speak about the change of encoding now.

The first attempt that has been taken into account to change the encoding of polynomials is the so-called 'sparse' form. Suppose a polynomial P has only few monomials with respect to its degree. The sparse encoding will be a number of vectors which specify the (non-zero) coefficient of every monomial appearing in P. For example, if $P = 2X^{15}Y^4 + 2X^7Y^3 - 3X^2 + 1$, it can be encoded by $P := (15, 4, 2); (7, 3, 2); (2, 0, -3); (0, 0, 1)$ instead of using a vector of $\binom{22}{2} = 231$ coordinates. There is a lot of theory and many algorithms that use this particular way of encoding polynomials. However, this encoding does not behave well under linear changes of coordinates in the sense that a 'short' polynomial in the sparse form can change into a 'long' one by a linear change of variables:

$$(X+Y)^{100} = \sum_{0 \leq i \leq 100} \binom{100}{i} X^i Y^{100-i}.$$

Another way to encode polynomials (the one we are going to study in these notes) has to do with the following idea:
Let P be the polynomial $P := (X+Y)^{100} - 1$. How come that we can define this polynomial so easily but it takes so much space to encode it for a machine?
The answer perhaps is that we are used to think of a polynomial as a 'formal expression' rather than a function that can be evaluated. But, as far as fields of characteristic zero are involved, polynomial functions can be considered just the same objects as polynomials. Therefore, if we define a polynomial function by defining its exact value at every point (that is to say, by means of describing how to evaluate it), we will be defining a polynomial. In the previous example, the polynomial P would be the only polynomial in $\mathbb{Q}[X, Y]$ such that, to evaluate it in a pair (x, y), you have to compute the sum of x and y to the 100-th power and subtract 1 from the result. This way of encoding a polynomial will be called a *straight-line program*:

Definition 5.1 *Let $X_1, \ldots, X_n$ be indeterminates over $\mathbb{Q}$ and let $R \in \mathbb{N}$. An element $\beta := (Q_1, \ldots, Q_R) \in \mathbb{Q}[X_1, \ldots, X_n]^R$ is a* **straight-line program** *(slp for short) if the elements Q_ρ satisfy one of the following two conditions:*

- *$Q_\rho \in \mathbb{Q} \cup \{X_1, \ldots, X_n\}$ or*
- *$\exists \rho_1, \rho_2 < \rho$ and $* \in \{+, -, \cdot, \div\}$ such that $Q_\rho = Q_{\rho_1} * Q_{\rho_2}$.*

We say β is **division-free** *if $Q_\rho = Q_{\rho_1} \div Q_{\rho_2} \Rightarrow Q_{\rho_2} \in \mathbb{Q} - \{0\}$.*

From now on, we are only going to deal with **division-free** slp's. Note that, in this case, each element Q_ρ is a polynomial in $\mathbb{Q}[X_1, \ldots, X_n]$. If $F \in \{Q_\rho \ / \ 1 \leq \rho \leq R\}$, we say that β *calculates* F.

There are several measures of complexity that can be taken into account when considering slp's. For example:

- The *total complexity* of β ($L(\beta)$) is the quantity of operations performed during the slp β (more precisely, it is the number of coordinates Q_ρ defined as the result of an operation between two previous coordinates.
- The *additive complexity* of β ($L_\pm(\beta)$) is the quantity of sums and subtractions performed during the slp.
- The *non-scalar complexity* of β ($L_{\mathbb{Q}}(\beta)$) is the number of products between two non-rational elements performed during the slp.

Given any polynomial $F \in \mathbb{Q}[X_1, \ldots, X_n]$ we will define its **total complexity** as

$$L(F) := \min\{L(\beta) \ / \ \beta \text{ is an slp computing } F\}.$$

We can respectively define $L_\pm(F)$ and $L_{\mathbb{Q}}(F)$.
It is easy to prove, for example, that for any polynomial F, $L(F) \leq \mathcal{O}((L_{\mathbb{Q}}(F))^2)$.

From now on, unless it is expressly stated, we will only deal consider the total complexity of an slp and of a polynomial and, for the sake of shortness, we will simply call it its complexity.

As an example, we are going to show a slp that calculates the polynomial $F(X) = 1 + X + X^2 + X^3 + \ldots + X^{2^j-1}$ efficiently. Of course, we can compute every power of X and then add them up, but it would yield an slp of complexity $2^{j+1} - 1$. Another slp computing F is the following:

$$\beta := \Big(1, X, X^2, X^4, \ldots, X^{2^{j-1}}, 1+X, 1+X^2, 1+X^4, \ldots, 1+X^{2^{j-1}},$$

$$(1+X)(1+X^2), (1+X)(1+X^2)(1+X^4), \ldots, \prod_{0 \leq i \leq 2^{j-1}} (1+X^i)\Big)$$

and $L(\beta) = 3j - 1$.

Another well-known example of short slp evaluating a polynomial is the Horner's rule:

$$a_0 + a_1X + a_2X^2 + \ldots + a_dX^d = (a_0 + X(a_1 + X(a_2 + X(\ldots(a_{d-1} + a_dX)\ldots))).$$

The length of this slp is $2d$ and it involves d products and d sums. It can be proved that the number of sums and the number of products involved in *any* slp computing this polynomial are bounded by d when the elements $X, a_0, \ldots, a_d$ are algebraically independent (see [1]).

When we are dealing with slp's to encode polynomials, there is a fundamental problem: the same polynomial may be encoded by means of many slp's. So, it is not straightforward to verify a polynomial identity.
Suppose you are given an slp of length β that evaluates a polynomial F in n variables of degree bounded by d. If you want to know whether $F \equiv 0$, a naive intent would be to interpolate F but it would take many points to do so, and the complexity would be again too large (within the same order of the dense representation of F) .
Another way to face the problem is to find a small particular set of points such that two polynomials of bounded length coincide if and only if they coincide in all these points. In this sense, we have the following

Theorem 5.2 *(see [13]) Let $\widetilde{W}(d,n,L)$ the set of polynomials in $\mathbb{Q}[X_1,\ldots,X_n]$ of degree bounded by d that can be calculated by means of an slp of length L. Let $\Gamma \subset \mathbb{Q}$ a set of $2L(1+d)^2$ elements. Then, there exists a set of points $\{\alpha_1,\ldots,\alpha_m\} \subset \Gamma^n$ with $m = 6(L+n)(L+n+1)$ satisfying*

$$F, G \in \widetilde{W}(d,n,L) \text{ such that } F(\alpha_i) = G_(\alpha_i)\ \forall\ 1 \leq i \leq m \Rightarrow F \equiv G.$$

The set $\{\alpha_1,\ldots,\alpha_m\}$ is called a set of *questors*. Unfortunately we do not know how to construct such a set within a reasonable cost. A way to avoid this problem is to consider probabilistic algorithms (we will discuss them later).

Another question we can ask is how many polynomials can be evaluated easily (that is to say, can be calculated by means of short slp's). The answer, again, is not very encouraging.

For fixed n, d and L, let's consider the set of all the polynomials $F \in \mathbb{Q}[X_1,\ldots,X_n]$ with $\deg(f) \leq d$ and non-scalar length $L_{\mathbb{Q}}(F) \leq L$.

Observe that each of these polynomials can be computed by a 'non-scalar' slp

$$\beta := (\beta_{-n+1},\ldots,\beta_0,\beta_1,\ldots,\beta_L)$$

where $\beta_{-n+i} = X_i$ $(1 \leq i \leq n)$ and, defining $\beta_{-n} := 1$,

$$\beta_k = \left(\sum_{-n\leq j\leq k-1} a_j^{(k)}.\beta_j)\right) . \left(\sum_{-n\leq j\leq k-1} b_j^{(k)}\beta_j\right).$$

Considering $a_j^{(k)}$ and $b_j^{(k)}$ $(1 \leq k \leq L;\ -n \leq j \leq k-1)$ as new variables, we can observe that any polynomial F_k that can be computed in the k-th step of β has as coefficients fixed polynomials in these new variables:

$$F_k = \sum_\alpha Q_\alpha^k(a,b)X^\alpha\ ,\ \text{with } Q_\alpha^k \in \mathbb{Q}[a_j^k, b_j^k]$$

for some a, b. So we have

Proposition 5.3 *For every L, $n \in \mathbb{N}$ there exist polynomials $Q_\alpha \in \mathbb{Z}[T_1,\ldots,T_m]$ with $m = (L+n)(L+n+1)$, $\alpha \in (\mathbb{N}_0)^n$ $|\alpha| \leq 2^L$, $\deg Q_\alpha \leq 2|\alpha|L$ such that for every $F \in \mathbb{Q}[X_1,\ldots,X_n]$ satisfying $L_{\mathbb{Q}}(F) \leq L$, then*

$$F = \sum_\alpha Q_\alpha(t)X^\alpha \text{ for some } t \in \mathbb{Q}^m.$$

Therefore, if $F := \sum_\alpha c_\alpha X^\alpha \in \mathbb{Q}[X_1,\ldots,X_n]$ has $\deg(f) \leq d$ and non-scalar length $L_{\mathbb{Q}}(F) \leq L$, considering it as the vector $(c_\alpha) \in \mathbb{Q}^{\binom{n+d}{n}}$, it turns out that $F \in Im(Q_\alpha\ :\ |\alpha| \leq d)$.

As a consequence, we have that, for fixed $d, n, L \in \mathbb{N}$, the set

$$W(n,d,L) := \overline{Im(Q_\alpha\ :\ \alpha \in \mathbb{N}_0,\ |\alpha| \leq d)} \subset \mathbb{C}^{\binom{n+d}{n}}$$

is a closed set that contains all the vectors of coefficients of polynomials $F \in \mathbb{Q}[X_1,\ldots,X_n]$ such that $\deg F \leq d$ and $L_{\mathbb{Q}} \leq L$.

A very important remark is that, as $W(n,d,L)$ is defined by means of a polynomial function using polynomials in $(L+n)(L+n+1)$ variables, its dimension is bounded by $\dim W(n,d,L) \leq$

$(L+n)(L+n+1)$ and, therefore, the smaller the L, the less the polynomials in $W(n,d,L)$; that is to say, most polynomials are difficult to evaluate.

Taking these last observations into account, one may wonder if it would be useful to deal with slp's when trying to solve polynomial equations. The answer is affirmative as we will see in the following sections.

5.1 A Fundamental Result

In [10], Giusti and Heintz, obtain a fundamental result using for the first time straight-line programs to solve a system of polynomial equations. In that paper, they can decide whether a given algebraic variety V is empty or not in **polynomial time** from the polynomials defining V. In fact, they can also find the dimension of V. The general idea is to compute, from the input polynomials, a particular variety Z, either zero-dimensional or empty, satisfying the following conditions:

- $V_0 \subset Z \subset V$ (that is to say, all the isolated points of V are in Z and all the points of Z are points in the variety.)
- It is 'easy' to decide whether Z is empty or not.

Note that, if we already know the variety V is either empty or has dimension 0, we can decide if it is empty by means of this result ($V = \emptyset \iff Z = \emptyset$).

Moreover, suppose we have the variety V defined by $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$. Generally, if it is not empty, when we cut it with a hyperplane H_1, we will obtain a variety $V \cap H_1$ of dimension $\dim V - 1$. Continuing this process with 'general' hyperplanes, we will have that, after $n+1$ steps, by reducing the dimension by one in each step, we get the empty set:

$$V \cap H_1 \cap \ldots \cap H_{n+1} = \emptyset.$$

So, we have that $V \cap H_1 \cap \ldots \cap H_n$ is either the empty set or a variety consisting only of isolated points and we are under the needed hypotheses to decide whether it is empty or not. If it is not empty, then $\dim V = n$. If it is empty, we consider the variety $V \cap H_1 \cap \ldots \cap H_{n-1}$ and repeat the process. After at most $n+1$ steps we will know the dimension of V.

The whole proof of the result by Giusti and Heintz is beyond the scope of these notes, but we are going to take into account some of the ideas used there. Moreover, we have not defined exactly what we meant by 'general' hyperplanes in the last paragraph: let's think that it means the whole construction works for almost every set of $n+1$ hyperplanes.

In the next section, we are going to see why it is 'easy' to decide whether the variety Z is empty or not.

5.2 An Old Way of Describing Varieties: the Shape Lemma

In [10], Giusti and Heintz use a particular way of defining zero-dimensional varieties which was already used by Kronecker (see [17]). This way of presenting the variety is called *a shape lemma presentation* or a *geometric resolution* of the variety.
The idea of this presentation is quite simple: Suppose we are given a zero-dimensional variety $Z \in \mathbb{C}^n$ defined by polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$ consisting of m points

$$x^{(1)} = (x_1^{(1)}, \ldots, x_n^{(1)}), \ldots, x^{(m)} = (x_1^{(m)}, \ldots, x_n^{(m)})$$

and suppose that their first coordinates are all distinct from one another. Therefore, we can obtain a polynomial $Q \in \mathbb{Q}[T]$ of degree m whose zeroes are exactly these first coordinates; namely

$$Q = \prod_{1 \leq i \leq m} (T - x_1^{(i)}).$$

Moreover, using interpolation, fixing an index j, $(2 \leq j \leq n)$, there exists a unique polynomial $P_j \in \mathbb{Q}[T]$ of degree bounded by $m-1$ such that $P_j(x_1^{(i)}) = x_j^{(i)}$ for every $1 \leq i \leq m$. Then, it is easy to see that

$$Z = \{x \in \mathbb{C}^n \ / \ Q(x_1) = 0 \wedge x_2 - P_2(x_1) = 0 \wedge \ldots \wedge x_n - P_n(x_1) = 0\}.$$

This parametric description of Z (note that all the coordinates are parameterized in function of x_1) has the additional property of telling us how many points are in Z (this number coincides with the degree of P_1).
The only inconvenient of this description is that we need the first coordinates of the points to be different from one another and this is not always the case. The way to solve this is to consider an affine linear form $\ell(X) = u_0 + u_1 X_1 + \ldots + u_n X_n$ in $\mathbb{Q}[X_1, \ldots, X_n]$ such that $\ell(x^{(i)})$ are all different from one another.
Now, we are able to define what we call a *geometric resolution* of a zero dimensional variety:

Definition 5.4 *Let $Z = \{x^{(1)}, \ldots, x^{(m)}\} \subset \mathbb{C}^n$ a zero dimensional variety defined by polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$. A geometric resolution of Z consists of an affine linear form $\ell(X) = u_0 + u_1 X_1 + \ldots + u_n X_n$ in $\mathbb{Q}[X_1, \ldots, X_n]$, and polynomials $Q, P_1, \ldots, P_n \in \mathbb{Q}[T]$ (where T is a new variable) such that:*

- *$\ell(x^{(i)}) \neq \ell(x^{(k)})$ if $i \neq k$ (in this case we say that ℓ is a primitive element of Z).*
- *$Q(T) = \prod_{1 \leq i \leq m} (T - \ell(x^{(i)}))$*
- *For $1 \leq i \leq n$, $\deg P_i \leq m-1$ and*

$$Z = \{(P_1(\xi), \ldots, P_n(\xi)) \ / \ \xi \in \mathbb{C} \text{ such that } Q(\xi) = 0\}.$$

As this description of Z is uniquely determined up to ℓ we call it the geometric resolution of Z associated to ℓ.

Although this definition is quite easy to understand, the problem underlying it is to find (given the zero-dimensional variety $Z \subset \mathbb{C}^n$ defined by polynomials $f_1, \ldots, f_s$) a proper linear form and the polynomials $Q, P_1, \ldots, P_n$ (note that our definition is based on our knowledge of the coordinates of the points in Z).
In [10] Giusti and Heintz do not find the exact geometric resolution of the isolated points of a variety V but they are able to find a linear form ℓ which separates the isolated points of V, a polynomial which annihilates over the specialization of ℓ in the isolated points of V and, by means of them, they found a variety Z either zero-dimensional or empty, satisfying $V_0 \subset Z \subset V$. The techniques they use are based on the regularity of the Hilbert function of a graduated ring and the introduction of a new variable to make a deformation.

For a detailed algorithmic construction of a geometric resolution of a zero-dimensional variety from polynomials defining it, see [16].

5.3 Newer Algorithms, Lower Bounds

The paper we have already mentioned ([10]) was a milestone in the development of algorithms solving polynomial equations. The main theorem they prove there is:

Theorem 5.5 *Given polynomials $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$ of degrees bounded by d in dense encoding defining an algebraic variety $V \subset \mathbb{C}^n$, there exists an algorithm that computes* $\dim V$ *within complexity $s^{\mathcal{O}(1)} d^{\mathcal{O}(n)}$.*

Note that this result allows us to answer the first question concerning a polynomial equation system (whether its set of solutions is empty or not) just by computing its dimension.

In some other works that came afterwards, by means of tools too sophisticated for the level of these notes, some of the problems stated above were solved within polynomial time.

In [11], given polynomials $f_1, \ldots, f_s \in \mathbb{Q}[X_1, \ldots, X_n]$ such that the variety they define is empty, a family of polynomials $g_1, \ldots, g_s \in \mathbb{Q}[X_1, \ldots, X_n]$ is constructed so that $1 = \sum_{1 \leq i \leq s} g_i . f_i$ holds. The polynomials $g_1, \ldots, g_s$ have degree bounded by $d^{\mathcal{O}(n^2)}$ and are obtained in an slp encoding. The complexity of the whole algorithm is $s^{\mathcal{O}(1)} d^{\mathcal{O}(n)}$ (compare with the end of Subsection 4.1).

In [6], by using duality theory, the same problem is re-considered and the improvement made is that the new polynomials $g_1, \ldots, g_s$ obtained have degree bounded by $d^{\mathcal{O}(n)}$.

A quantifier elimination algorithm using *slp* was obtained in [19]. The main result there is more general than the one we stated above but, adapted to our case it would essentially mean that the elimination stated before can be done in polynomial time in the size of the input.

We can also mention algorithms for the equidimensional decomposition of varieties that can be performed in polynomial time in the size of the input (see for example [18] and [14]). However, both of these algorithms, are probabilistic (see Section 7.1 for a brief account on probabilistic algorithms).

6 The Newton-Hensel Method

The use of slp's as a way of encoding polynomials made it possible to adapt algorithmically a very well-known concept, the Newton-Hensel method, which can be seen as a version of the implicit function theorem.
Let $T_1, \ldots, T_m, X_1, \ldots, X_n$ be indeterminates over a field $\mathbb{Q}$. Given $t \in \mathbb{C}^n$, $T - t$ will represent the vector $(T_1 - t_1, \ldots, T_n - t_n)$.
Let $f_1, \ldots, f_n \in \mathbb{Q}[T, X]$ be polynomials. We will denote by f the vector of polynomials $(f_1, \ldots, f_n)$, by Df the Jacobian matrix of f with respect to the indeterminates X and by J_f its determinant.

Lemma 6.1 *Let $f_1, \ldots, f_n \in \mathbb{Q}[T, X]$ and let $(t, \xi) \in \mathbb{C}^m \times \mathbb{C}^n$ such that*

$$f_1(t, \xi) = 0, \ldots, f_n(t, \xi) = 0 \text{ and } Jf(t, \xi) \neq).$$

Then, there exists a vector of formal series $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_n) \in \mathbb{C}[[T - t]]^n$ such that:

- $f_1(T, \mathcal{R}) = 0, \ldots, f_n(T, R) = 0$
- $\mathcal{R}(t) := (\mathcal{R}_1(t), \ldots, \mathcal{R}_n(t)) = \xi$.

Sketch of the Proof. Given $f(X) = (f_1(X,T), \ldots, f_n(X,T))$, we define the Newton-Hensel operator associated to it as

$$N_f(X)^t := X^t - Df(X)^{-1}.f(X)^t.$$

Note that $Jf(X)$ is not the zero polynomial (from our hypothesis, $Jf(t,\xi) \neq 0$). We define the following succession of rational functions:

$$\begin{cases} \mathcal{R}^{(0)} := \xi \\ \mathcal{R}^{(k)} := N_f(\mathcal{R}^{(k-1)}) = N_f^k(\xi) \ \text{ for } \ k \in \mathbb{N} \end{cases}$$

The first thing to take into account is whether we can define this succession (that is to say, if we are not trying to divide by zero) but this fact can be inductively proved supposing $\mathcal{R}^k(t) = \xi$.

The following conditions are fulfilled (this can be proved recursively):

- $f_j(T, \mathcal{R}^{(k)}) \in (T-t)^{2^k} \subset \mathbb{C}[[T-t]]$ for every $1 \leq j \leq n$.
- $\mathcal{R}_i^{(k-1)} - \mathcal{R}_i^{(k)} \in (T-t)^{2^k} \subset \mathbb{C}[[T-t]]$ for every $1 \leq i \leq n$

where $(T-t)$ indicates the ideal in $\mathbb{C}[[T-t]]$ generated by $T_1 - t_1, \ldots, T_m - t_m$.
Therefore, the successions $(\mathcal{R}_i^{(k)})_{k \in \mathbb{N}}$ are convergent ($1 \leq i \leq n$) and the vector of their limits $\mathcal{R} := (\mathcal{R}_1, \ldots, \mathcal{R}_n)$ is the vector we are looking for. □

There are two main features to be taken into account when considering Newton-Hensel algorithm. The first one is that the precision we obtain to get our series is exponential (that is to say, to obtain the series we are looking for up to degree ℓ we only have to apply $\log_2 \ell$ steps of our iteration) and this implies it is very quick. The second one is that the Newton-Hensel procedure deals essentially with slp's. In fact, an algorithmic statement of the Newton-Hensel method is the following

Lemma 6.2 *(see [7]) Under the same hypotheses and notations we used in Lemma 6.1, suppose the polynomials $f_1, \ldots, f_n$ have degree bounded by d and are given by an slp of length L. Let $\kappa \in \mathbb{N}$, then there exists an slp of length $O(kd^2n^7L)$ which evaluates polynomials $g_1^{(\kappa)}, \ldots, g_n^{(\kappa)}, h^{(\kappa)} \in \mathbb{Q}[T][X]$ with $h^{(\kappa)}(t,\xi) \neq 0$ which represent the numerators and the denominator of the rational functions obtained in the κ-th iteration of the Newton-Hensel operator.*

Just to show how this works, we are going to discuss an example briefly.

Suppose we are given generic polynomials $f_1, \ldots, f_n$ of degree $d_1, \ldots, d_n$ in a group of several variables T and $X_1, \ldots, X_n$:

$$f_1(X,T) = \sum_{|\alpha| \leq d_1} T_\alpha^{(1)} X^\alpha$$
$$\ldots$$
$$f_n(X,T) = \sum_{|\alpha| \leq d_n} T_\alpha^{(n)} X^\alpha$$

(note that we are introducing a new variable T for every coefficient of f_i ($1 \leq i \leq n$).)
Consider the system $V := \{X_1^{d_1} - 1, \ldots, X_n^{d_n} - 1\}$. Of course, we know all the roots of this second system: they are vectors of roots of unity. Therefore we can consider that we are under the conditions needed to apply Lemma 6.1 because we have a certain instance of T (let call it t) and of X (a vector of roots of unity we will call ξ) that satisfy the needed hypotheses (it is easy

to see that in this instance, $Jf(t,\xi) \neq 0$). Then, by applying the Newton-Hensel algorithm we can approximate vectors of power series in $T-t$ which will be roots of the original system and we can do it as precisely as we want.
We will have then $\prod_{1\leq i\leq n} d_i$ different (approximations of) vectors of power series that should be all the roots of the original system in $\overline{\mathbb{C}(T)}$ (it can be seen that the system we are dealing with is equidimensional of dimension zero when we think of T as a set of parameters and Bezout's theorem states that the number of solutions is bounded by $\prod_{1\leq i\leq n} d_i$).

Moreover, any linear form $L := a_1X_1 + \ldots + a_nX_n$ that separates the different points in V, must separate the different points in our original system.
Suppose that, from a root $v \in V$, we obtain the root $\mathcal{R}_v \in \mathbb{C}[[T-t]]^n$ of the original system. Then

$$\prod_{v\in V}(Y - L(\mathcal{R}_v))$$

is a polynomial in $\mathbb{Q}[[T-t]][Y]$ that vanishes at every point $\mathcal{R}_v$. In fact, this polynomial is the polynomial of minimal degree defining the image of our original variety under the morphism of varieties

$$\overline{\mathbb{Q}(T)}^n \to \overline{\mathbb{Q}(T)}$$
$$w \to L(w)$$

As our original variety is definable with polynomials in $\mathbb{Q}(T)[X]$, this polynomial we obtain must be in $\mathbb{Q}(T)[Y]$ and therefore, by multiplying it by a fixed polynomial $R \in \mathbb{Q}[T]$ we obtain a polynomial in $M \in \mathbb{Q}[T][Y]$ satisfying the following:

$$M(T, a_1X_1 + \ldots + a_nX_n) \in (f_1, \ldots, f_n)$$

(here we are using that the ideal the polynomials $f_1, \ldots, f_n$ define is radical).
Therefore, except for the values of t lying in a hypersurface, we have a non-zero polynomial $\widetilde{M}(t,Y) \in \mathbb{Q}[Y]$ which specialized in the linear form L vanishes over the zeroes of V and this is a fundamental step we mentioned before (see Sections 5.1 and 5.2).
Of course a lot of work has to be done to succeed in finding this polynomial. For example one should know somehow up to what precision the Newton-Hensel algorithm is needed, but this is just an example of how things may work.

7 Other Trends

In this last section, I would like to comment on some ideas involved in algorithmical procedures which were mentioned before.

7.1 Probabilistic algorithms

Sometimes our algorithms depend on the choice of an object satysfing certain conditions (a linear form separating points, a point where a polynomial does not vanish, etc). These choices may be very expensive from the algorithmic point of view. Think of a polynomial f in n indeterminates of degree d. If you want to get a vector v such that $f(v) \neq 0$ we have to check at many points. Sometimes, they may even involve a procedure we do not know how to accomplish (for example,

we know we have to look for a point that is not a root of certain polynomial of bounded degree, but we do not know exactly the polynomial). To avoid this, one can choose a random point v to go on. Of course, this may probably lead to an error. Then, a probabilistic algorithm would be an algorithm that 'generally' performs the task we want accurately, but with a bounded probability of error.
Most algorithms involving slp's can be considered as probability algorithms if we do not know an adequate family of questors for the kind of slp's involved. In this case, if we want to decide whether an slp represent the 0 polynomial or not, we just choose a random point and evaluate the slp in it. If the result is not zero, we are sure that the polynomial is not the zero polynomial but if it is zero, we can suppose probably that the polynomial is the zero one.
A clear example of this is the following: We have a non-empty variety V and we want to compute its dimension. We cut it with a random hyperplane and consider what happens. Suppose that this intersection is empty. We would assume that the original variety is of dimension 0. It is generally the case, but if we are unlucky and the original variety was lying in a hyperplane parallel to the one we chose, our deduction would be false.

In the probabilistic algorithms we consider, the generic condition a random point should satisfy is that it is not a zero of a given polynomial $f \in \mathbb{Q}[x_1, \ldots, x_n]$ of bounded degree. The random point we choose has integer coordinates taken from a finite set of $\mathbb{N}$ big enough. The estimation of the probability of success is done by means of the following well-known result (see [21]):

Lemma 7.1 *$R \subset \mathbb{N}$ be a finite subset. Let $f \in \mathbb{Q}[x_1, \ldots, x_n] - \{0\}$ be a polynomial. Then, for random choices of elements $a_1, \ldots, a_n \in R$, we have that*

$$\mathrm{Prob}(f(a_1, \ldots, a_n) = 0) \leq \frac{\deg f}{\#R}.$$

For example, both the equidimensional decomposition algorithms mentioned in 5.3 are probabilistic.

7.2 Non-general algorithms

In Section 5, we have mentioned that a possible way to avoid the high complexities involved in dense encoding was to design specific algorithms that would not work for every polynomial system but for some of them. This is already being done, in the sense that some of the algorithms being produced in Computer Algebra may be general but work better (have lower complexity) in special cases. That is why the measure of complexity sometimes takes into account other invariants (not only the degree, quantity and number of variables of the polynomials involved). Roughly speaking, the new invariants involved have to do with the geometry of the varieties involved (that is the *semantic* features of the problem) and not with the way the variety is presented (the *syntactic* ones). For a further discussion about this theme, see [8].

References

[1] Bürgisser, P., Clausen, M. and Amin Shokrollahi, M. , *Algebraic Complexity Theory* , Springer (1997).

[2] Chang, C. C. and Keisler, H. J., *Model Theory*, North Holland (1973).

[3] Chistov, A. L. and Grigor'ev, D. Yu., *Subexponential-time solving systems of algebraic equations I, II*, Steklov Mathematical Institute, Lenigrad Department, LOMI Preprints E-9-83, 0e-10-c83 (1983).

[4] Cox D., Little J. and O'Shea, D., *Ideals, Varieties and Algorithms*, Springer-Verlag (1992).

[5] Fitchas, N. and Galligo, A., *Nullstellensatz effective et conjecture de Serre (Théorème de Quillen-Suslin) pour le calcul formel*, Math. Nachr. **149** (1990), 231-253.

[6] Fitchas, N., Giusti, M. and Smietanski, F., *Sur le complexité du théorème des zéros*, Approximation and Optimization **8** (1995), 274-329.

[7] Giusti, M., Hägele, K., Heintz, J., Montaña, J. L., Morais, J. E. and Pardo, L. M., *Lower bounds for Diophantine approximation*, J. Pure Appl. Algebra **117 & 118** (1997) 277-317.

[8] Giusti, M., Heintz, J., Morais, J. E., Morgenstern, J. and Pardo, L. M., *Straight-line programs in geometric elimination theory*, J. Pure Appl. Algebra **124** (1998) 101-146.

[9] Giusti, M. and Heintz, J., *Algorithmes -disons rapides- pour la décomposition d'une variété algébrique en composantes irréductibles et équidimensionelles*, Progr. Math. **94**, Birkäuser (1991),169-193.

[10] Giusti, M. and Heintz, J., *La détermination des points isolés et de la dimension d'une variété peut se faire en temps polynomial*, Sympos. Math., Vol XXXIV (1993), 216-256.

[11] Giusti, M., Heintz, J. and Sabia, J., *On the effectivity of effective Nullstellensätze*, Comput. Complexity **3** (1993) 56-95.

[12] Hartshorne, R., *Algebraic Geometry* Springer (1977).

[13] Heintz, J. and Schnorr, C. P., *Testing polynomials which are easy to compute*, Monographie **30** de l'Enseignement Mathématique (1982), 237-254.

[14] Jeronimo, G. and Sabia, J., *Effective equidimensional decomposition of affine varieties*, J. Pure Appl. Algebra **169** 2-3 (2002), 229-248.

[15] Kollár, J., *Sharp effective Nullstellensatz*, J. Amer. Math. Soc **1** No. 1 (1988), 963-975.

[16] Krick, T. and Pardo, L. M., *A computational method for Diophantine approximation*, Prog. Math. **143** (1996), 193-254.

[17] Kronecker, L., *Grundzüge einer arithmetischen Theorie de algebraischen Grössen*, J. reine angew. Math. **92** (1882), 1-122.

[18] Lecerf, G., *Computing an equidimensional decomposition of an variety by means of geometric resolutions*, Proc. ISSAC'2000, ACM (2000), 209-216.

[19] Puddu, S. and Sabia, J., *An effective algorithm for quantifier elimination over algebraically closed fields using straight-line programs*, J. Pure Appl. Algebra **129** (1998), 173-200.

[20] Sabia, J. and Solernó, P., *Bounds for traces in complete intersections and degrees in the Nullstellensatz*, AAECC J. **6** (1995), 353-376.

[21] Schwartz, J. T., *Fast probabilistic algorithms for verification of polynomial identities*, J. ACM **27** (1980),701-717.

[22] Shafarevich, I.R., *Basic Algebraic Geometry*, Springer-Verlag (1974).

[23] Sombra, M. , *Bounds for the Hilbert function of polynomial ideals and for the degrees in the Nullstellensatz*, J. Pure Appl. Algebra **117 & 118** (1997), 565-599.