1

On the Complexity of Non Universal Polynomial Equation Solving: Old and New Results

C. Beltrán

Departamento de Matemáticas
Universidad de Cantabria
Santander, Spain
e-mail: beltranc@unican.es

L. M. Pardo

Departamento de Matemáticas
Universidad de Cantabria
Santander, Spain
e-mail: luis.pardo@unican.es

Abstract

These pages summarize some results on the efficiency of polynomial equation solving. We focus on semantic algorithms, i.e., algorithms whose running time depends on some intrinsic/semantic invariant associated with the input data. Both computer algebra and numerical analysis algorithms are discussed. We show a probabilistic and positive answer to Smale's 17th problem. Estimates of the probability distribution of the condition number of singular complex matrices are also exhibited.

1.1 Introduction

These pages summarize some results on upper and lower complexity bounds in Elimination Theory. They are a revision of the program stated in Pardo (1995).

We focus on *Efficient Polynomial Equation Solving*. This is one of the challenges in the recent history of Computational Mathematics. Two main frameworks in scientific computing deal with this problem. Following different approaches, symbolic/algebraic computing and numerical analysis developed their own techniques for solving polynomial equations. We survey statements of both approaches. New results are contained in Sections 1.4 and 1.5.

Multivariate Polynomial Equation Solving is a central topic both in Computational Mathematics and Computational Algebraic Geometry

(Elimination Theory in nineteenth century terminology). Its origin goes back to work of Sturm, Hermite, Cayley, and Sylvester, among others. Elimination Theory consists of the preparation of input data (polynomial equations and inequalities) to answer questions involving quantifiers. This approach also underlies Kronecker (1882), Hilbert (1890) and further developments in Algebraic Geometry. A central problem in Elimination Theory is the following:

Problem 1 (Hilbert's Nullstellensatz) *Design an efficient algorithm that performs the following task:*
Given a system of multivariate polynomial equations

$$f_1, \ldots, f_s \in \mathbb{C}[X_1, \ldots, X_n],$$

decide whether the following algebraic variety is empty or not:

$$V(f_1, \ldots, f_s) := \{x \in \mathbb{C}^n \ : \ f_i(x) = 0, 1 \leq i \leq s\}.$$

Here the term efficient refers to computational complexity. In the words of Traub & Werschultz (1998): *"computational complexity is a measure of the intrinsic computational resources required to solve a mathematical problem"*. Computational resources are measured in terms of a computational model or computational device that performs the corresponding algorithm that solves the problem. Intrinsic here means that we measure resources required by the problem and not the concrete algorithm. Hence, computational complexity is the design and analysis of an optimal algorithm (in terms of computational resources) that solves a mathematical problem.

The notion of computational resource requirements has been present in the mathematical literature for many years, although not always in an explicit form. For instance, we cite Galois who explicitly described computational requirements in his *Mémoire sur la Résolubilité des Équations par Radicaux*. Galois wrote: *"En un mot, les calculs sont impracticables"*. Galois had developed an algorithm that decided whether a univariate polynomial equation was solvable by radicals, but he realized that the computational complexity required by his procedure was excessive. The phrase thus means that he declined to perform calculations. In fact, he had discovered a central subject in computational complexity: *Intractability*.

In Galois' time, neither the notion of algorithm nor that of a complexity measure had been established. This relevant step in the history of mathematics was made around 1933. The works of Gödel, Church and Turing established the notion of algorithm which in later years lead to the existence of computers. We note that Turing's work and his machine concept

of algorithm also became the standard pattern for computational complexity. In these pages, we shall measure computational resources in terms of Turing machines as much as is possible.

Computational resources are measured as functions of the input length. The input length is the time we need to write down the data. The (running) time function is the function that relates input length and running time under a concrete computational model.

Intractability is one of the frustrating aspects of computational complexity studies. A mathematical problem is intractable if the computational resources required to solve it are so excessive that there is no hope of solving the problem in practice. Observe that intractability is independent of the algorithm we design. For example, mathematical problems whose running time are at least exponential in the input length are naturally intractable. These are called *exponential problems* and there is no hope of solving them in any real or future computer. The reason is that this exponential time requirement is intrinsic to the problem and not to the concrete algorithm or computer.

Tractable problems are those mathematical problems whose time function is bounded by a polynomial of the input length. Between tractable and intractable problems there lies a large number of problems for which it is not known as yet whether they are tractable. We call them the Boundary of Intractability (cf. Garey & Johnson (1979)). Hilbert's Nullstellensatz lies in this boundary. This simply means that no-one has yet designed a tractable algorithm that solves Hilbert's Nullstellensatz, and it also means that no-one has yet proved that this problem is intractable. That is, it is not known whether there is an algorithm that solves **HN** in running time which depends polynomially on the number of variables.

There are several strategies for studying the computational complexity of Hilbert's Nullstellensatz. We classify them in two main groups: *syntactical* and *semantical*. Although these pages are mainly concerned with semantical strategies, we shall sketch some of the syntactical achievements in the study of **HN**.

Syntactical strategies are characterized by the fact that polynomials are considered as lists of coefficients (dense encoding) in certain vector spaces. They are then treated as vectors and linear algebra methods are applied to answer questions (mainly those involving quantifiers).

Historically, the first syntactical algorithm for **HN** goes back to Hilbert and his student Hermann (cf. Hermann (1926)). Hilbert and Hermann reduced **HN** to the consistency of a system of linear equations.

Hilbert's Nullstellensatz (Hilbert (1890)) states that *given a list of polynomials* $f_1, \ldots, f_s \in \mathbb{C}[X_1, \ldots, X_n]$ *of degree at most* d*, the complex*

algebraic variety they define $V(f_1,\dots,f_s)\subseteq\mathbb{C}^n$ is empty if an only if there are polynomials $g_1,\dots,g_s\in\mathbb{C}[X_1,\dots,X_n]$ such that the following equality holds:

$$1 = g_1 f_1 + \cdots + g_s f_s. \tag{1.1}$$

Identities such as (1.1) are called *Bézout Identities*.

From Hermann's work, we know that there is a function $D(d,n)$ which depends only on the number of variables and the maximum of the degrees, such that the following equivalence holds:

The variety $V(f_1,\dots,f_s)\subseteq\mathbb{C}^n$ is empty if and only if there exist polynomials $g_1,\dots,g_s$ in $\mathbb{C}[X_1,\dots,X_n]$ of degree at most $D(d,n)$ satisfying identity (1.1).

Let us observe that Hermann's bound $D(d,n)$ reduces **HN** to the consistency question of a system of linear equations. The unknowns are the coefficients of the (possibly existing) polynomials $g_1,\dots,g_s$ occurring in (1.1). The linear equations are determined by linear functions in the coefficients of the input polynomials $f_1,\dots,f_s$. This approach reduces **HN** to the problem of deciding consistency of the linear system given by (1.1) involving

$$s\binom{D(d,n)+n}{n}$$

variables and equations. Its running time is obviously polynomial in this quantity. Hence, sharp upper bounds for the function $D(d,n)$ also imply sharp upper complexity bounds for this approach to solving **HN**. Studies on sharp upper bounds for $D(d,n)$ are called *Effective Nullstellensätze*. We cite Brownawell (1987), Caniglia, Galligo & J. Heintz (1988), Kollár (1988), Berenstein & Yger (1991, 1991a), Krick & Pardo (1996), Hägele, Morais, Pardo & M. Sombra (2000), Krick, Pardo & Sombra (2001) and their references. The known bounds for $D(d,n)$ can be summarized by the following inequalities:

$$d^{n-1}\leq D(d,n)\leq d^n.$$

Thus, this approach is neither efficient nor applicable since the time complexity is of order

$$\binom{d^n+n}{n}\approx d^{n^2}.$$

For example, deciding consistency of a system of cubic polynomial equations in 20 variables by this method requires deciding consistency of a system of more than 3^{400} linear equations in a similar number of variables. This is intractable in any actual or future computer.

In Hägele, Morais, Pardo & Sombra (2000) a simply exponential time algorithm (time of order d^n) to compute Bézout identities was shown, although the technique used in that paper is not syntactical but semantical.

A second syntactical strategy to deal with **HN** is due to rewriting techniques. The most frequently used rewriting method is that of the standard/Gröbner basis algorithms. Since the works Hironaka (1964) and Buchberger (1965), a huge list of references has been produced (cf. for instance Becker & Weispfenning (1993), Cox, Little & O'Shea (1997), Mora (2003), Vasconcelos (1998) and references therein). Most of these references discuss algorithms that compute Gröbner bases of an ideal. This strategy has also been fruitful in terms of implementations. Gröbner basis algorithmics is a standard primitive implemented in most computer algebra packages (Maple, Magma or Mathematica, for example). Most efficient implementations are due to Faugère (the *FGb* series). This approach has a serious drawback in terms of computational complexity. Since Mayr & Meyer (1982), we know that computing with Gröbner bases is exponential space complete and this is even worse than the running time of methods based on Effective Nullstellensätze. Computing with Gröbner bases involving more than 15 variables is not yet available. Thus, purely syntactical Gröbner bases techniques do not seem to be the best methods of dealing with **HN**.

A third syntactical strategy uses the underlying concepts of Structural Complexity. Namely, problems are classified into complexity classes and the study of the complexity of a problem consists in locating the appropriate class where this problem is complete. In Blum, Shub & Smale (1989), the authors proved that **HN** is complete in the class $\mathbf{NP}_{\mathbb{C}}$ of nondeterministic polynomial time under the abstract model of complex Turing machines (cf. also Blum, Cucker, Shub & Smale (1998)). Other authors studied the complexity of **HN** within the more realistic Turing machine framework. In Koiran (1996) (see also Rojas (2001, 2003)) the author proved that **HN** belongs to the complexity class **PH** (polynomial hierarchy).

Nevertheless, all these syntactical strategies seem to forget that we are dealing with geometric objects (algebraic varieties) and regular mappings (polynomials viewed as functions and not as mere lists of coefficients). Algebraic varieties and regular mappings are mathematical objects rich in terms of semantic invariants. They have been studied for years in an attempt to describe their topological, geometrical and arithmetical properties. These studies have generated a large number of semantic invariants that must be related to computational complexity. This idea of relating semantical invariants to complexity is not completely new. In fact, semantical invariants of geometrical objects have been used to show lower complexity

bounds for computational problems (see, for example, Montaña, Morais & Pardo (1996) and references therein). The converse problem was to design an algorithm that solves **HN** in time which depends polynomially on some semantical invariants of the input list of multivariate polynomials. This was achieved by the TERA experience. In fact, the TERA experience was more a current of thought than a research project, that was active during the nineties. Some of its achievements will be described in Section 1.2.

Somewhere between syntactical and semantical strategies, we may find "sparse" elimination techniques as in Sturmfels (1996) and references therein. However, we do not discuss sparse elimination here.

The rest of the chapter is structured as follows. In Section 1.2 we present an overview of some of the achievements of the TERA experience. In Section 1.3 we discuss an exponential lower time bound for universal algorithms in Elimination Theory. In Section 1.4 we show a positive answer to Smale's 17th Problem. Finally, in Section 1.5 we show sharp upper bounds for the probability distribution of the condition number of singular matrices.

1.2 Semantic Algorithms

In the middle nineties, the notion of *semantic algorithms in Elimination Theory* was introduced. Two works initiated this new generation of algorithms : Pardo (1995) and Giusti, Heintz, Morais & Pardo (1995). The first paper established a program, whereas the second one exhibited the first example of a semantical algorithm for Elimination Theory. This program was achieved in the series of papers Giusti, Heintz, Morais, Morgenstern & Pardo (1998), Giusti, Hägele, Heintz, Montaña, Morais & Pardo (1997), Giusti, Heintz, Morais & Pardo (1997). These works became the basis of the research experience called TERA. This section is devoted to a brief sketch of some of these achievements.

First of all, we reformulate Hilbert's Nullstellensatz in the following form.

Problem 2 *Design an efficient algorithm that performs the following task: Given a list of polynomials $f_1, \ldots, f_s, g \in \mathbb{C}[X_1, \ldots, X_n]$ of degree at most d, decide whether the polynomial g vanishes at some point of the algebraic variety $V(f_1, \ldots, f_s) \subseteq \mathbb{C}^n$.*

This is the usual formulation of elimination polynomials (like resultants and discriminants) in classical Elimination Theory. This is also the usual formulation of **NP**–complete problems (cf. Heintz & Morgenstern (1993) or Pardo (1995) and references therein). Note that all **NP**–complete problems are particular instances of Problem 2 above.

In this formulation, the role played by g and the list of $f_1, \ldots, f_s$ seems to be different.

From a list of polynomials like $f_1, \ldots, f_s$ we want to compute some *information* concerning the variety $V := V(f_1, \ldots, f_s)$ of its common zeros. The information we compute is expected to be used to answer further questions involving the variety V. This information is commonly called a *solution* of the input list of polynomials $f_1, \ldots, f_s$. For instance, in Problem 2, a solution of $f_1, \ldots, f_s$ should be used to decide whether a new polynomial g vanishes at some point in V. The way we choose to represent this information in a computer may be called the *encoding* of the solution variety V.

Obviously, different questions will condition the way we represent the information on the variety V in a computer. Hence, different notions of solution lead to different kinds of algorithms and different encodings of algebraic varieties. In Section 1.4 we recall the Shub–Smale notion of solution (approximate zeros) whose potentiality is still unexplored.

The proposal of TERA consisted in the design and analysis of a semantic algorithm that performs the following task:

From an input list $f_1, \ldots, f_s$, the algorithm outputs a description of the solution variety $V(f_1, \ldots, f_s)$.

This algorithm must satisfy two main properties:

- Its running time should be bounded by some intrinsic/semantic quantity that depends on the input list.
- Its output must contain sufficient information to answer any kind of elimination question like the one described in Problem 2.

These two properties lead to a notion of solution that we briefly sketch here. It is called *Kronecker's encoding of an affine algebraic variety* (cf. Kronecker (1882)).

Let $f_1, \ldots, f_i \in \mathbb{C}[X_1, \ldots, X_n]$ be a sequence of polynomials defining a radical ideal $(f_1, \ldots, f_i)$ of codimension i. Let $V := V(f_1, \ldots, f_i) \subseteq \mathbb{C}^n$ be the complex algebraic variety of dimension $n-i$ given by its common zeros. A *Kronecker encoding* of V is a birational isomorphism of V with some complex algebraic hypersurface in some affine complex space of dimension $n - i + 1$.

Technically, this is expressed as follows. Firstly, let us assume that the variables $X_1, \ldots, X_n$ are in Noether position with respect to the variety V. Namely, we assume that the following is an integral ring extension:

$$\mathbb{C}[X_1, \ldots, X_{n-i}] \hookrightarrow \mathbb{C}[X_1, \ldots, X_n]/(f_1, \ldots, f_i).$$

Let $u := \lambda_{n-i+1} X_{n-i+1} + \cdots + \lambda_n X_n \in \mathbb{Q}[X_1, \ldots, X_n]$ be a linear form

in the dependent variables $\{X_{n-i+1}, \ldots, X_n\}$. A Noether's normalization and the linear mapping u define a linear projection:

$$\mathcal{U} : \mathbb{C}^n \longrightarrow \mathbb{C}^{n-i+1} \ : \ (x_1, \ldots, x_n) \longmapsto (x_1, \ldots, x_{n-i}, u(x_1, \ldots, x_n)) .$$

Let $\mathcal{U} \mid_V : V \longrightarrow \mathbb{C}^{n-i+1}$ be the restriction of the projection $\mathcal{U}$ to the variety V. The image set of the projection $\mathcal{U} \mid_V$ is a complex hypersurface H_u in $\mathbb{C}^{n-i+1}$. Let us denote by $\chi_u \in \mathbb{C}[X_1, \ldots, X_{n-i}, T]$ the minimal equation of H_u. The polynomial χ_u is called the elimination polynomial of u with respect to V.

The linear form u is called a *primitive element* if and only if the projection $\mathcal{U} \mid_V$ defines a birational isomorphism of V with H_u.

A Kronecker solution of the system of polynomial equations $f_1 = 0, \ldots, f_i = 0$ consists of a description of the Noether normalization, the primitive element u, the hypersurface H_u and a description of the inverse of the birational isomorphism, i.e., a description of $(\mathcal{U} \mid_V)^{-1}$. Formally, this list of items can be given as follows:

- The list of variables in Noether position $X_1, \ldots, X_n$ (which includes a description of the dimension of V). It is just a regular matrix that defines a linear change of coordinates that puts the variables in Noether position.
- The primitive element $u := \lambda_{n-i+1} X_{n-i+1} + \cdots + \lambda_n X_n$ given by its coefficients in $\mathbb{Z}$ (or any other computable subfield of $\mathbb{C}$).
- The minimal equation χ_u of the hypersurface H_u.
- A description of $(\mathcal{U} \mid_V)^{-1}$. This description can be given by the following list of polynomials:
  - A nonzero polynomial $\rho \in \mathbb{C}[X_1, \ldots, X_{n-i}]$.
  - A list of polynomials $v_j \in \mathbb{C}[X_1, \ldots, X_{n-i}, T]$, $n - i + 1 \leq j \leq n$.

  These polynomials must satisfy the equality:

  $$(\mathcal{U} \mid_V)^{-1}(x, t) = \left(x_1, \ldots, x_{n-i}, \rho^{-1}(x) v_{n-i+1}(x, t), \ldots, \rho^{-1}(x) v_n(x, t)\right) ,$$

  for all $x := (x_1, \ldots, x_{n-i}) \in \mathbb{C}^{n-i}$, $t \in \mathbb{C}$, such that $(x, t) \in H_u$, $\rho(x) \neq 0$.

In 1882, Kronecker conceived an iterative procedure for solving multivariate systems of equations $F := [f_1, \ldots, f_n]$ defining zero–dimensional complex varieties. Kronecker's idea can be sketched in the following terms:

First, the procedure starts with system $[f_1]$ and it "solves" the equidimensional variety of codimension one $V(f_1) \subseteq \mathbb{C}^n$. Then the procedure runs iteratively: From a Kronecker encoding of $V(f_1, \ldots, f_i)$, the procedure "eliminates" the polynomial f_{i+1} to obtain a Kronecker encoding of the "next" variety $V(f_1, \ldots, f_{i+1})$. Proceed until $i = n$ is reached.

This iterative procedure has two main drawbacks, which can be explained in the following terms:

- First of all, the storage problem arising with the encoding of the intermediate polynomials. The polynomials χ_u, ρ and v_j are polynomials of high degree (eventually of degree d^i) involving $n-i+1$ variables. Thus, to compute with them, the procedure has to handle all their coefficients, which amounts to
$$\binom{d^i+n-i+1}{n-i+1}$$
in number, For example, for $i := n/2$ the procedure must save more than $d^{n^2/4}$ coefficients. Handling such polynomials also requires a time complexity of similar order. This does not seem to be more efficient than the original treatment based on the Effective Nullstellensätze (cf. Section 1.1).
- Secondly, Kronecker's iterative procedure introduces a nesting of interpolation procedures. This nesting is demanded by the iterative process. Every time the procedure computes a new set of variables in Noether position, the procedure makes a recursive call of previously computed objects. This increases the time complexity function to $d^{O(n^2)}$.

The procedure was therefore forgotten by contemporary mathematicians and is hardly mentioned in the literature of Algebraic Geometry. Macaulay quotes Kronecker's procedure in Macaulay (1916) and so does König (1903). But both of them thought that this procedure would require excessive running time to be efficient, and so references to it have progressively vanished from the literature. Traces of this procedure can be found spread over the Algebraic Geometry literature without giving the required reference to it. For example, Kronecker's notion of solution was used in Zariski (1995) to define a notion of dimension for algebraic varieties, claiming that it was also used in the same form by Severi and others.

In Giusti, Heintz, Morais & Pardo (1995) and Pardo (1995), Kronecker's approach for solving was rediscovered without previous knowledge of this ancestor. These two works were able to overcome the first drawback (space problem of representation) of the previous methods. The technical trick was the use of a data structure coming from semi–numerical modeling: straight–line programs. This idea of representing polynomials by programs evaluating them goes back to previous work of the same research group (such as Giusti, Heintz (1991, 1993) or Krick & Pardo (1996), see also the references given in Pardo (1995)).

To overcome the second drawback (nesting), the authors introduced a method based on nonarchimedean Newton's operator. The approximate zeros in the corresponding nonarchimedean basin of attraction were called *Lifting Fibers* in Giusti, Hägele, Heintz, Morais, Montaña & Pardo (1997)

solving the problem of nesting of interpolation procedures by Hensel's Lemma (also called the Implicit Mapping Theorem).

Unfortunately, Giusti, Hägele, Heintz, Morais, Montaña & Pardo (1997) introduced (for the Lifting Fibers) running time requirements which depend on the heights of the intermediate varieties in the sense of Bost, Gillet & Soulé (1994) or Philippon (1991, 1994, 1995). This drawback was finally overcome in Giusti, Heintz, Morais & Pardo (1997), where integer numbers were represented by straight–line programs and the following result was finally established:

Theorem 1.1 *(Giusti, Heintz, Morais & Pardo (1997)) There exists a bounded error probability Turing machine M which performs the following task: Given a system of multivariate polynomial equations $F := (f_1, \ldots, f_n)$, satisfying the following properties*

- $\deg(f_i) \leq d$ *and* $ht(f_i) \leq h$ *for* $1 \leq i \leq n$ *(h is the bit length of the coefficients),*
- *the ideals $(f_1, \ldots, f_i)$ are radical ideals of codimension i in the ring $\mathbb{Q}[X_1, \ldots, X_n]$ for $1 \leq i \leq n-1$,*
- *the variety $V(f_1, \ldots, f_n) \subseteq \mathbb{C}^n$ is a zero–dimensional complex algebraic variety,*

then the machine M outputs a Kronecker solution of the variety

$$V(f_1, \ldots, f_n).$$

The running time of the machine M is polynomial in the quantities

$$\delta(F),\ n,\ h,\ d,\ L,$$

where $\delta(F)$ is the maximum of the degrees of the intermediate varieties (in the sense of Heintz (1983)), namely

$$\delta(F) := \max\{\deg(V(f_1, \ldots, f_i)) \ : \ 1 \leq i \leq n-1\},$$

and L is the input length in any natural encoding of multivariate polynomials.

It must be said that the coefficients of the polynomials involved in a Kronecker solution of the variety $V(f_1, \ldots, f_n)$ are given by straight–line programs. However, the complexity estimates for the Turing machine M are independent of the height.

The quantity $\delta(F)$ becomes a kind of condition number for symbolic methods to solve systems of multivariate polynomial equations by Kronecker's deformation technique.

After Giusti, Heintz, Morais, & Pardo (1997), several new authors got involved in the TERA experience, with several technical improvements,