Linear Solving for Sign Determination

Daniel Perrucci

Abstract

We give a specific method to solve with quadratic complexity the linear systems arising in known algorithms to deal with the sign determination problem, both in the univariate and multivariate setting. In particular, this enables us to improve the complexity bound for sign determination in the univariate case to $O(sd^2 \log^3 d)$, where s is the number of polynomials involved and d is a bound for their degree. Previously known complexity results involve a factor of $d^{2.376}$.

1 Introduction

Let R be a real closed field. The sign determination problem is a basic problem in computational real algebraic geometry. It consists in determining the sign conditions realized by a finite list $\mathcal{P} = P_1, \dots, P_s$ of polynomials in $\mathrm{R}[X_1, \dots, X_k]$ on a finite set $Z \subset \mathrm{R}^k$ which is not known explicitly, but defined as the zero set of an ideal in $\mathrm{R}[X_1, \dots, X_k]$. These sign conditions are said to be the feasible sign conditions for $\mathcal{P}$ on Z.

Most algorithms dealing with this problem rely on two main ingredients. The first one consists in the computation of the Tarski-query (also known as Sturm-query) for Z of a polynomial $P \in \mathrm{R}[X_1, \dots, X_k]$, which is the number of elements in Z where P takes a positive value minus the number of elements in Z where P takes a negative value. The second one consists in solving a linear system which relates some previously computed Tarski-queries for Z with the number of elements in Z satisfying certain sign conditions for $\mathcal{P}$.

As an example, the naive approach to solve the sign determination problem can be described as follows: first compute the Tarski-query for Z of each of the 3^s polynomials of type $P_1^{e_1} \dots P_s^{e_s}$, $e_i = 0, 1, 2$ for $i = 1, \dots, s$, and then solve the particular linear system of size $3^s \times 3^s$ which relates the vector formed with these known quantities and the one formed with the unknown cardinalities of the sets $Z \cap \{P_1 \sigma_1 0, \dots, P_s \sigma_s 0\}$, $\sigma_i \in \{<, =, >\}$ for $i = 1, \dots, s$. If $m = \#Z$, note that at most m sign conditions will be realized on Z, and then at most m of the coordinates of the solution will be different from 0.

In [2], the exponential complexity arising from the number of Tarski-query computations and the resolution of a linear system of size $3^s \times 3^s$ in the approach above is avoided. This is achieved by means of a recursive algorithm in which the list $\mathcal{P}$ is divided into two sublists,

the number of points in Z satisfying each feasible sign condition for each sublist is computed, and then this information is combined. Such combination is obtained by computing at most m^2 Tarski-queries and solving a linear system of size at most $m^2 \times m^2$.

In [6], [3] and [1, Chapter 10], the methods in [2] are further developed. In [6], an algorithm is given where the number of points in Z satisfying each feasible sign condition for the list $P_1, \ldots, P_i$ is computed sequentially for $i = 1, \ldots, s$, taking into account that, at each step, each feasible sign condition for $P_1, \ldots, P_{i-1}$ may be extended in at most 3 ways. To deal with the addition of the polynomial P_i to the considered list, at most $2m$ Tarski-queries are computed and a linear system of size at most $3m \times 3m$ is solved. In [3], a more explicit way to choose the polynomials whose Tarski-query is to be computed is given. In [1, Chapter 10], also the feasible sign conditions for P_i on Z are computed at step i, in order to discard beforehand some non-feasible sign conditions for $P_1, \ldots, P_i$ on Z extending feasible sign conditions for $P_1, \ldots, P_{i-1}$ on Z.

The most efficient algorithms presently known to deal with the sign determination problem follow the approach described in the paragraph above. Depending on the setting, the Tarski-queries may be computed in different ways, taking a different number of operations in the field R, or in a proper domain D containing the coefficients of the polynomials in $\mathcal{P}$ and the polynomials defining the set Z. On the other hand, as treated with general methods, the linear solving part takes $O(m^{2.376})$ operations in $\mathbb{Q}$ ([4]). In the univariate case, the Tarski-queries can be computed so fast that the complexity of solving the linear systems leads the overall complexity. Following the analysis in ([3, Section 3]) for the univariate case, if d is a bound for the degree of the polynomials in $\mathcal{P}$ and a given polynomial $P_0 \in \mathrm{R}[X_1]$ having Z as its set of roots in R, at step i, for $i = 1, \ldots, s$, the complexity of the Tarski-query computations is $O(md \log m \log^2 d)$; then the complexity of the whole sign determination algorithm is $O(s(md \log m \log^2 d + m^{2.376}))$. When m is unknown and its value is bounded by d, this complexity bound becomes $O(sd^{2.376})$.

In this paper we design a specific method (Algorithm SDlinsolve, Section 3), to solve with a better complexity bound the linear systems involved in the sign determination problem. Our main result is the following:

Theorem 1 *The linear systems arising in the sign determination algorithm can be solved within $O(m^2)$ operations in $\mathbb{Q}$.*

The algorithm to solve such systems is based on a factorization of the inverse of the matrix defining the considered linear system, obtained by following a block Gauss elimination procedure (Proposition 6). Some of the matrices in this factorization are known explicitly, and some of them are described in terms of inverses of other matrices defining also linear systems of the considered family, but with smaller size. This allows the algorithm to proceed recursively, which is a key fact to obtain the desired complexity bound.

The algorithm presented here can be used as a subroutine both in the univariate and the multivariate setting. In the univariate case, it also allows us to improve the complexity

bound for the whole sign determination algorithm. By replacing by $O(m^2)$ the complexity of the linear solving steps in the complexity analysis done before, we obtain the following corollary, which is in fact the main motivation for this work:

Corollary 2 *Given $P_0, P_1, \ldots, P_s \in \mathrm{R}[X_1]$, $P_0 \not\equiv 0$, $\deg P_i \leq d$ for $i = 0, \ldots, s$, the feasible sign conditions for $P_1, \ldots, P_s$ on $\{P_0 = 0\}$ (and the number of elements in $\{P_0 = 0\}$ satisfying each of these sign conditions) can be computed within $O(sd^2 \log^3 d)$ operations in* R*. Moreover, if P_0 has m roots in* R*, this can be done within $O(smd \log(m) \log^2(d))$ operations in* R*.*

In [5], the need for a quadratic algorithm to solve the sign determination problem in the univariate case is apparent. In this work, a *probabilistic* algorithm to determine all the feasible sign conditions on $\mathbb{R}^k$ for a given list of polynomials is presented. This algorithm computes a *geometric resolution*, which is a univariate parametric description, of a finite set of sample points; in this way, all the feasible sign conditions on $\mathbb{R}^k$ for the given list of polynomials is obtained by computing the sign of these polynomials at this finite set. In this reduction to the univariate case, the degree of the polynomials obtained equals the Bézout number $\delta \sim d^k$ of some auxiliary polynomial systems, and the complexity of the algorithm depends quadratically on δ. Following [3, Section 3], the complexity to obtain from this *geometric resolution* all the feasible sign conditions contains a factor of $\delta^{2.376}$, which increases the overall complexity. As a direct application of Corollary 2, we have that the complexity of the algorithms in [5, Theorems 17 and 26] to determine all the feasible sign conditions for multivariate polynomials depends quadratically on δ, improving the best known complexity bound for *probabilistic* algorithms solving this problem.

2 Preliminaries and Notation

We will use mainly the notation in [1, Chapter 10]. In this reference, the approach described in the introduction is followed with the minor difference that the polynomials $P_1, \ldots, P_s$ are introduced one at each step from back to front; therefore, the notation is adapted to this order.

For $i = 1, \ldots, s$, we call $\mathcal{P}_i$ the list $P_i, \ldots, P_s$. For $P \in \mathrm{R}[X_1, \ldots, X_k]$, we denote by $c(P = 0, Z), c(P > 0, Z)$ and $c(P < 0, Z)$ the number of elements in Z satisfying the condition $P = 0, P > 0$ and $P < 0$ respectively. For $\sigma \in \{0, 1, -1\}^{\mathcal{P}_i}$, we denote by $c(\sigma, Z)$ the number of elements x in Z satisfying $\mathrm{sign}(P_j(x)) = \sigma(P_j)$ for $j = i, \ldots, s$. For any list $\Sigma = \sigma_1, \ldots, \sigma_l$ of elements in $\{0, 1, -1\}^{\mathcal{P}_i}$, we denote by $c(\Sigma, Z)$ the vector whose components are $c(\sigma_1, Z), \ldots, c(\sigma_l, Z)$. The Tarski-query of a polynomial P for Z is the number

$$\mathrm{TaQ}(P, Z) = c(P > 0, Z) - c(P < 0, Z).$$

For a list $\mathcal{Q}$ in $\mathrm{R}[X_1, \ldots, X_k]$, $\mathrm{TaQ}(\mathcal{Q}, Z)$ is the vector formed by the Tarski-queries for Z of the polynomials in $\mathcal{Q}$. We denote by $\hat{\sigma}$ the element of $\{0, 1, -1\}^{\mathcal{P}_{i+1}}$ obtained from σ by

deleting the coordinate corresponding to P_i and by $\hat{\Sigma}$ the list $\hat{\sigma}_1, \ldots, \hat{\sigma}_l$. Note that $\hat{\Sigma}$ might contain repeated elements even if all the elements in Σ are different.

As explained in the introduction, if the sign determination algorithm is at step i ($i = s, \ldots, 1$), we are given a particular list $\Sigma = \sigma_1, \ldots, \sigma_r$ of elements in $\{0, 1, -1\}^{\mathcal{P}_i}$ with $\sigma_1 <_{\text{lex}} \cdots <_{\text{lex}} \sigma_r$ ($0 \prec 1 \prec -1$) containing, maybe properly, all the feasible sign conditions for $\mathcal{P}_i$ on Z and we are to compute the exact list of feasible sign conditions for $\mathcal{P}_i$ on Z and the number of elements in Z satisfying each of these sign conditions. The inequality $r \leq 3m$ holds at every step.

We divide the given list Σ into 12 ordered sublists taking into account the number of repetitions in $\hat{\Sigma}$ and how the sign conditions in $\hat{\Sigma}$ are extended in Σ: for $\emptyset \neq B \subset \{0, 1, -1\}$ and $b \in B$, the list Σ^b_B is composed by those $\sigma \in \Sigma$ such that $\sigma(P_i) = b$ and the set

$$\{b' \in \{0, 1, -1\} \mid \exists \sigma' \in \Sigma \text{ such that } \sigma' \text{ extends } \hat{\sigma} \text{ and } \sigma'(P_i) = b'\}$$

equals B. For simplicity, if $B = \{b_1, \ldots, b_l\}$, we write $\Sigma^b_{b_1,\ldots,b_l}$ for $\Sigma^b_{\{b_1,\ldots,b_l\}}$. We also write Σ_0, Σ_1 and Σ_{-1} for Σ^0_0, Σ^1_1 and Σ^{-1}_{-1} respectively. In addition, since $\hat{(\Sigma^b_{b_1,\ldots,b_l})}$ is the same list for every $b \in B$, we denote by $\hat{\Sigma}_{b_1,\ldots,b_l}$ any such list.

We also divide the list Σ into 3 ordered sublists as follows: $\Sigma_{(1)}$ is the list obtained by merging $\Sigma_0, \Sigma_1, \Sigma_{-1}$, $\Sigma^0_{0,1}$, $\Sigma^0_{0,-1}$, $\Sigma^{-1}_{1,-1}$ and $\Sigma^0_{0,1,-1}$, $\Sigma_{(2)}$ is the list obtained by merging lists $\Sigma^1_{0,1}, \Sigma^{-1}_{0,-1}, \Sigma^1_{1,-1}$ and $\Sigma^1_{0,1,-1}$ and $\Sigma_{(3)}$ is the same list as $\Sigma^{-1}_{0,1,-1}$. In this way, $\Sigma_{(1)}$ contains one extension of every element in $\hat{\Sigma}$, $\Sigma_{(2)}$ contains one extension of every element repeated at least twice in $\hat{\Sigma}$, $\Sigma_{(3)}$ contains one extension of every element repeated three times in $\hat{\Sigma}$ and $\hat{\Sigma}_{(1)}, \hat{\Sigma}_{(2)}$ and $\hat{\Sigma}_{(3)}$ do not have repeated elements.

Consider also the list $\text{Ada}(\Sigma)$ of elements in $\{0, 1, 2\}^{\mathcal{P}_i}$ (which represents a list of multidegrees) defined recursively by:

$$\begin{cases} 0 & \text{if } i = s \text{ and } r = 1, \\ 0, 1 & \text{if } i = s \text{ and } r = 2, \\ 0, 1, 2 & \text{if } i = s \text{ and } r = 3, \\ 0 \times \text{Ada}(\hat{\Sigma}_{(1)}),\ 1 \times \text{Ada}(\hat{\Sigma}_{(2)}),\ 2 \times \text{Ada}(\hat{\Sigma}_{(3)}) & \text{if } i < s. \end{cases}$$

We denote by $\mathcal{P}_i^{\text{Ada}(\Sigma)}$ the list of polynomials formed by the polynomials in $\mathcal{P}_i$ raised to the multidegrees in $\text{Ada}(\Sigma)$.

To illustrate the introduced notation, we include the following example.

Example 3 *Suppose* $s = 3, i = 1$ *and* $\Sigma = (0,0,0), (0,1,0), (1,0,0), (1,0,-1), (1,1,0), (1,1,-1), (-1,0,0), (-1,0,-1), (-1,1,-1), (-1,-1,1), (-1,-1,-1)$. *Then we have:*

- Σ_0 *and* Σ_1 *are empty lists and* $\Sigma_{-1} = (-1,-1,1), (-1,-1,-1)$*;*

- $\Sigma^0_{0,-1}$ *and* $\Sigma^{-1}_{0,-1}$ *are empty lists and* $\Sigma^0_{0,1} = (0,1,0)$; $\Sigma^1_{0,1} = (1,1,0)$; $\Sigma^1_{1,-1} = (1,0,-1),(1,1,-1)$ *and* $\Sigma^{-1}_{1,-1} = (-1,0,-1),(-1,1,-1)$*;*
- $\Sigma^0_{0,1,-1} = (0,0,0)$; $\Sigma^1_{0,1,-1} = (1,0,0)$ *and* $\Sigma^{-1}_{0,1,-1} = (-1,0,0)$*;*
- $\Sigma_{(1)} = (0,0,0), (0,1,0), (-1,0,-1), (-1,1,-1), (-1,-1,1), (-1,-1,-1)$*;* $\Sigma_{(2)} = (1,0,0),(1,0,-1),(1,1,0),(1,1,-1)$ *and* $\Sigma_{(3)} = (-1,0,0)$*;*
- $\mathrm{Ada}(\Sigma) = (0,0,0), (0,0,1), (0,0,2), (0,1,0), (0,1,1), (0,2,0), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (2,0,0)$.
- $\mathcal{P}_i^{\mathrm{Ada}(\Sigma)} = 1,\ P_3,\ P_3^2,\ P_2,\ P_2P_3,\ P_2^2,\ P_1,\ P_1P_3,\ P_1P_2, P_1P_2P_3, P_1^2$.

For a list $A = \alpha_1, \ldots, \alpha_{l_1}$ of elements in $\{0,1,2\}^{\mathcal{P}_i}$ and a list $\Sigma' = \sigma'_1, \ldots, \sigma'_{l_2}$ of elements in $\{0,1,-1\}^{\mathcal{P}_i}$, we denote by $\mathrm{Mat}(A,\Sigma')$ the $\mathbb{Z}^{l_1 \times l_2}$ matrix defined by

$$\mathrm{Mat}(A,\Sigma')_{j_1 j_2} = {\sigma'_{j_2}}^{\alpha_{j_1}},$$

for $j_1 = 1, \ldots, l_1$ and $j_2 = 1, \ldots, l_2$, with the understanding that $0^0 = 1$. The main property of the matrix above is the following:

Proposition 4 *(See [3, Sections 2 and 3] or [1, Proposition 10.65]). For every list of sign conditions* Σ*, the matrix* $\mathrm{Mat}(\mathrm{Ada}(\Sigma),\Sigma)$ *is invertible. Moreover, if* Σ *contains all the feasible sign conditions for* $\mathcal{P}_i$ *on* Z*, then:*

$$\mathrm{Mat}(\mathrm{Ada}(\Sigma),\Sigma)\, c(\Sigma, Z) = \mathrm{TaQ}(\mathcal{P}_i^{\mathrm{Ada}(\Sigma)}, Z). \tag{1}$$

When convenient, for a matrix M with rows indexed by a list A of multidegrees and columns indexed by a list Σ' of sign conditions, and for any sublists A' of A and Σ'' of Σ', we will denote by $M_{A'}$, $M_{\Sigma''}$ and $M_{A',\Sigma''}$ the submatrices obtained from M by taking only the rows in A', only the columns in Σ'', and only the rows in A' and the columns in Σ'' respectively. We will use a similar notation for vectors whose coordinates are indexed by a list of multidegrees or a list of sign conditions.

3 The specific method for linear solving

Note that a different order in Σ would lead to a permutation of columns in the matrix $\mathrm{Mat}(\mathrm{Ada}(\Sigma),\Sigma)$ and the elements of the vector $c(\Sigma, Z)$. To explain our method in a simpler way, we will suppose that we change the order in Σ in such a way that we find first the elements in $\Sigma_{(1)}$, then those in $\Sigma_{(2)}$ and finally those in $\Sigma_{(3)}$. Nevertheless, this change of order is not actually necessary in the execution of the linear solving method given here.

If Σ is a list of sign conditions on a single polynomial, we have that either $r = 1$; $r = 2$ and $\Sigma = 0, 1$; $r = 2$ and $\Sigma = 0, -1$; $r = 2$ and $\Sigma = 1, -1$ or $r = 3$. Depending on which of these conditions holds, $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)$ is one of the following matrices:

$$\left(1 \right), \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix}.$$

If Σ is a list of sign conditions on many polynomials, consider the matrices $M_1 = \mathrm{Mat}(\mathrm{Ada}(\hat{\Sigma}_{(1)}), \hat{\Sigma}_{(1)})$, $M_2 = \mathrm{Mat}(\mathrm{Ada}(\hat{\Sigma}_{(2)}), \hat{\Sigma}_{(2)})$ and $M_3 = \mathrm{Mat}(\mathrm{Ada}(\hat{\Sigma}_{(3)}), \hat{\Sigma}_{(3)})$. Then, $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)$ is the matrix:

$$\left(\begin{array}{c|c|c} M_1 & M_1' & M_1'' \\ \hline X & \tilde{M}_2 & -M_2' \\ \hline Y & Z & M_3 \end{array}\right)$$

where:

- M_1' is the matrix formed by the columns of M_1 corresponding to sign conditions in $\hat{\Sigma}_{0,1}, \hat{\Sigma}_{0,-1}, \hat{\Sigma}_{1,-1}$ and $\hat{\Sigma}_{0,1,-1}$,
- M_1'' is the matrix formed by the columns of M_1 corresponding to sign conditions in $\hat{\Sigma}_{0,1,-1}$,
- $\tilde{M}_2$ is the matrix obtained from M_2 by multiplying by -1 the columns corresponding to sign conditions in $\hat{\Sigma}_{0,-1}$,
- M_2' is the matrix formed by the columns of M_2 corresponding to sign conditions in $\hat{\Sigma}_{0,1,-1}$,
- $X = \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_{(1)})$, $Y = \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_{(1)})$ and $Z = \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_{(2)})$.

Remark 5 • *The only non-zero columns in matrices X and Y are those corresponding to sign conditions in Σ_1, Σ_{-1} and $\Sigma_{1,-1}^{-1}$.*

- *The following relations are satisfied:*

$$X_{\Sigma_{1,-1}^{-1}} = -(M_2)_{\hat{\Sigma}_{1,-1}}, \qquad Y_{\Sigma_{1,-1}^{-1}} = Z_{\Sigma_{1,-1}^{1}}, \qquad Z_{\Sigma_{0,1,-1}^{1}} = M_3.$$

- *Since* $\hat{\Sigma}_{(3)}$ *is included in* $\hat{\Sigma}_{(2)}$, $\mathrm{Ada}(\hat{\Sigma}_{(3)})$ *is included in* $\mathrm{Ada}(\hat{\Sigma}_{(2)})$ *and then we have:*

$$X_{1\times \mathrm{Ada}(\hat{\Sigma}_{(3)}),\Sigma_1} = Y_{\Sigma_1}, \qquad X_{1\times \mathrm{Ada}(\hat{\Sigma}_{(3)}),\Sigma_{-1}} = -Y_{\Sigma_{-1}},$$

$$X_{1\times \mathrm{Ada}(\hat{\Sigma}_{(3)}),\Sigma^{-1}_{1,-1}} = -Y_{\Sigma^{-1}_{1,-1}}.$$

The rearrangement of columns of $\mathrm{Mat}(\mathrm{Ada}(\Sigma),\Sigma)$ according to the sublists $\Sigma_{(1)},\Sigma_{(2)}$ and $\Sigma_{(3)}$, allows us to follow a block Gauss elimination procedure to obtain a factorization of the inverse of $\mathrm{Mat}(\mathrm{Ada}(\Sigma),\Sigma)$ in terms of the matrices $N_1,\ldots,N_9$ we introduce below.

$$N_1 = \left(\begin{array}{c|c|c} M_1^{-1} & 0 & 0 \\ \hline 0 & I_2 & 0 \\ \hline 0 & 0 & I_3 \end{array}\right),\ N_2 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline -X & I_2 & 0 \\ \hline -Y & 0 & I_3 \end{array}\right),\ N_3 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline 0 & M_2^{-1} & 0 \\ \hline 0 & 0 & I_3 \end{array}\right),$$

$$N_4 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline 0 & \tilde{I}_2 & 0 \\ \hline 0 & 0 & I_3 \end{array}\right),\ N_5 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline 0 & I_2 & 0 \\ \hline 0 & -\tilde{Z} & I_3 \end{array}\right),\ N_6 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline 0 & I_2 & 0 \\ \hline 0 & 0 & M_3^{-1} \end{array}\right),$$

$$N_7 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline 0 & I_2 & 0 \\ \hline 0 & 0 & \frac{1}{2}I_3 \end{array}\right),\ N_8 = \left(\begin{array}{c|c|c} I_1 & 0 & 0 \\ \hline 0 & I_2 & I_2' \\ \hline 0 & 0 & I_3 \end{array}\right),\ N_9 = \left(\begin{array}{c|c|c} I_1 & -I_1' & -I_1'' \\ \hline 0 & I_2 & 0 \\ \hline 0 & 0 & I_3 \end{array}\right),$$

where I_1, I_2 and I_3 denote the identity matrices which size is the length of $\Sigma_{(1)},\Sigma_{(2)}$ and $\Sigma_{(3)}$ respectively and, if the columns of I_1, I_2 and I_3 are indexed with $\hat{\Sigma}_{(1)},\hat{\Sigma}_{(2)}$ and $\hat{\Sigma}_{(3)}$, then:

- $\tilde{I}_2$ is the matrix obtained from I_2 by multiplying by -1 the columns corresponding to sign conditions in $\hat{\Sigma}_{0,-1}$ and by $\frac{1}{2}$ the columns corresponding to sign conditions in $\hat{\Sigma}_{1,-1}$,

- $\tilde{Z}$ is the matrix obtained from Z by multiplying by 0 the columns corresponding to sign conditions in $\Sigma^1_{1,-1}$,
- I'_2 is the matrix formed by the columns of I_2 corresponding to sign conditions in $\hat{\Sigma}_{0,1,-1}$,
- I'_1 is the matrix formed by the columns of I_1 corresponding to sign conditions in $\hat{\Sigma}_{0,1}, \hat{\Sigma}_{0,-1}, \hat{\Sigma}_{1,-1}$ and $\hat{\Sigma}_{0,1,-1}$,
- I''_1 is the matrix formed by the columns of I_1 corresponding to sign conditions in $\hat{\Sigma}_{0,1,-1}$.

Proposition 6 *The matrix* $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)^{-1}$ *equals the product* $N_9 \ldots N_1$.

Proof: First, note that

$$M_1^{-1} M'_1 = I'_1, \qquad M_1^{-1} M''_1 = I''_1, \qquad M_2^{-1} M'_2 = I'_2.$$

Because of the first item of Remark 5 we conclude that

$$X I''_1 = 0, \qquad Y I''_1 = 0,$$

and using the first and second items of Remark 5, we conclude that

$$-X I'_1 + \tilde{M}_2 = \dot{M}_2, \qquad -Y I'_1 + Z = \tilde{Z}, \qquad Z I'_2 = M_3,$$

where $\dot{M}_2$ is the matrix obtained from M_2 by multiplying by -1 the columns corresponding to sign conditions in $\hat{\Sigma}_{0,-1}$ and by 2 the columns corresponding to sign conditions in $\hat{\Sigma}_{1,-1}$. With all these relations, the proof can be done by simple computation of the product $N_9 \ldots N_1 \mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)$.

□

This factorization of $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)^{-1}$ leads to the following recursive algorithm:

Algorithm: SDlinsolve

Input: a list $\Sigma = \sigma_1, \ldots, \sigma_r$ of sign conditions, a vector $v \in \mathbb{Q}^r$.

Output: $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)^{-1} v$.

Procedure:

0. Initialize i as the size of the tuples in Σ.
1. If $i = s$, return $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)^{-1} v$.
2. If $i < s$:
    0. Initialize $c = v$.

1. $c_{\Sigma_{(1)}} = \mathrm{SDlinsolve}(\hat{\Sigma}_{(1)}, c_{\Sigma_{(1)}})$.
2. $c_{\Sigma_{(2)}} = c_{\Sigma_{(2)}} - \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_1) c_{\Sigma_1} - \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_{-1}) c_{\Sigma_{-1}} - \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_{1,-1}^{-1}) c_{\Sigma_{1,-1}^{-1}}$;
   $c_{\Sigma_{(3)}} = c_{\Sigma_{(3)}} - \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_1) c_{\Sigma_1} - \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_{-1}) c_{\Sigma_{-1}} - \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_{1,-1}^{-1}) c_{\Sigma_{1,-1}^{-1}}$.
3. $c_{\Sigma_{(2)}} = \mathrm{SDlinsolve}(\hat{\Sigma}_{(2)}, c_{\Sigma_{(2)}})$.
4. $c_{\Sigma_{0,-1}^{-1}} = -c_{\Sigma_{0,-1}^{-1}}$;
   $c_{\Sigma_{1,-1}^{1}} = \frac{1}{2} c_{\Sigma_{1,-1}^{1}}$.
5. $c_{\Sigma_{(3)}} = c_{\Sigma_{(3)}} - \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_{0,1}^{1}) c_{\Sigma_{0,1}^{1}} - \mathrm{Mat}(2 \times \mathrm{Ada}(\hat{\Sigma}_{(3)}), \Sigma_{0,-1}^{-1}) c_{\Sigma_{0,-1}^{-1}} - M_3 c_{\Sigma_{0,1,-1}^{1}}$.
6. $c_{\Sigma_{(3)}} = \mathrm{SDlinsolve}(\hat{\Sigma}_{(3)}, c_{\Sigma_{(3)}})$.
7. $c_{\Sigma_{(3)}} = \frac{1}{2} c_{\Sigma_{(3)}}$.
8. $c_{\Sigma_{0,1,-1}^{1}} = c_{\Sigma_{0,1,-1}^{1}} + c_{\Sigma_{(3)}}$.
9. $c_{\Sigma_{0,1}^{0}} = c_{\Sigma_{0,1}^{0}} - c_{\Sigma_{0,1}^{1}}$;
   $c_{\Sigma_{0,-1}^{0}} = c_{\Sigma_{0,-1}^{0}} - c_{\Sigma_{0,1}^{-1}}$;
   $c_{\Sigma_{1,-1}^{-1}} = c_{\Sigma_{1,-1}^{-1}} - c_{\Sigma_{1,-1}^{1}}$;
   $c_{\Sigma_{0,1,-1}^{0}} = c_{\Sigma_{0,1,-1}^{0}} - c_{\Sigma_{0,1,-1}^{1}} - c_{\Sigma_{(3)}}$.
10. Return c.

We can give now the proof of Theorem 1:

Proof: Let us prove that the algorithm above solves the linear system (1) within $2r^2$ operations in $\mathbb{Q}$. The correctness of the algorithm follows since, for $j = 0, \ldots, 9$, after Step 2.j we have computed $c = N_j \ldots N_1 v$.

To bound the number of operations done by this algorithm, we proceed by induction on i. If $i = s$, the result follows, since the inverse of the 5 possible matrices $\mathrm{Mat}(\mathrm{Ada}(\Sigma), \Sigma)$ is pre-computed and the product by v takes $r(2r-1)$ operations in $\mathbb{Q}$.

Suppose now $i < s$. For $\emptyset \neq B = \{b_1, \ldots, b_l\} \subset \{0, 1, -1\}$ denote by $r_{b_1,\ldots,b_l}$ the size of the list $\Sigma^b_{b_1,\ldots,b_l}$ for any $b \in B$. Using the inductive hypothesis, the number of operations in each step is bounded in the following way:

2.1. $2(r_0 + r_1 + r_{-1} + r_{0,1} + r_{0,-1} + r_{1,-1} + r_{0,1,-1})^2$.

2.2. $2(r_{0,1} + r_{0,-1} + r_{1,-1} + 2r_{0,1,-1})(r_1 + r_{-1} + r_{1,-1})$.

2.3. $2(r_{0,1} + r_{0,-1} + r_{1,-1} + r_{0,1,-1})^2$.

2.4. $r_{0,-1} + r_{1,-1}$.

2.5. $2r_{0,1,-1}(r_{0,1} + r_{0,-1} + r_{0,1,-1})$.

2.6. $2r_{0,1,-1}^2$.

2.7. $r_{0,1,-1}$.

2.8. $r_{0,1,-1}$.

2.9. $r_{0,1} + r_{0,-1} + r_{1,-1} + 2r_{0,1,-1}$.

Since the sum of all these numbers is always lower than or equal to $2r^2 = 2(r_0 + r_1 + r_{-1} + 2r_{0,1} + 2r_{0,-1} + 2r_{1,-1} + 3r_{0,1,-1})^2$, the result follows taking into account that the inequality $r \leq 3m$ holds at every step.

□

Remark 7 *The third item of Remark 5 implies that Step 2.2 can be replaced in the following way:*

2.2.' $w = \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_1)c_{\Sigma_1}$;
$w' = \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_{-1})c_{\Sigma_{-1}}$;
$w'' = \mathrm{Mat}(1 \times \mathrm{Ada}(\hat{\Sigma}_{(2)}), \Sigma_{1,-1}^{-1})c_{\Sigma_{1,-1}^{-1}}$;
$c_{\Sigma_{(2)}} = c_{\Sigma_{(2)}} - w - w' - w''$;
$c_{\Sigma_{(3)}} = c_{\Sigma_{(3)}} - w_{1\times\mathrm{Ada}(\hat{\Sigma}_{(3)})} + w'_{1\times\mathrm{Ada}(\hat{\Sigma}_{(3)})} + w''_{1\times\mathrm{Ada}(\hat{\Sigma}_{(3)})}$.

which takes a smaller number of operations than Step 2.2.

References

References

[1] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, second edition, 2006.

[2] Michael Ben-Or, Dexter Kozen, and John Reif. The complexity of elementary algebra and geometry. *J. Comput. System Sci.*, 32(2):251–264, 1986. 16th annual ACM-SIGACT symposium on the theory of computing (Washington, D.C., 1984).

[3] John Canny. Improved algorithms for sign determination and existential quantifier elimination. *Comput. J.*, 36(5):409–418, 1993.

[4] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9(3):251–280, 1990.

[5] Gabriela Jeronimo, Daniel Perrucci, and Juan Sabia. On sign conditions over real multivariate polynomials. To appear in *Discrete Comput. Geom.* DOI: 10.1007/s00454-009-9200-4.

[6] Marie-Françoise Roy and Aviva Szpirglas. Complexity of computation on real algebraic numbers. *J. Symbolic Comput.*, 10(1):39–51, 1990.