A Stabilized Normal Form Algorithm for Generic Systems of Polynomial Equations

Simon Telen, Marc Van Barel*

April 12, 2018

Abstract

We propose a numerical linear algebra based method to find the multiplication operators of the quotient ring $\mathbb{C}[x]/I$ associated to a zero-dimensional ideal I generated by n $\mathbb{C}$-polynomials in n variables. We assume that the polynomials are generic in the sense that the number of solutions in $\mathbb{C}^n$ equals the Bézout number. The main contribution of this paper is an automated choice of basis for $\mathbb{C}[x]/I$, which is crucial for the feasibility of normal form methods in finite precision arithmetic. This choice is based on numerical linear algebra techniques and it depends on the given generators of I.

1 Introduction

Consider the following problem. Given n polynomials $f_1, \ldots, f_n \in k[x_1, \ldots, x_n]$ with k an algebraically closed field, find all the points $x \in k^n$ where they all vanish: $f_1(x) = \ldots = f_n(x) = 0$. Here, we will work over the complex numbers $k = \mathbb{C}$. The ring of all polynomials in the n variables $x_1, \ldots, x_n$ with coefficients in $\mathbb{C}$ is denoted by $\mathbb{C}[x_1, \ldots, x_n]$. For short, we will denote $x = (x_1, \ldots, x_n)$ and an element $f \in \mathbb{C}[x]$ can be written as

$$f = \sum_{\alpha \in \mathbb{Z}^n_{\geq 0}} c_\alpha x^\alpha$$

where we used the short notation $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. The *support* $S(f)$ of f is defined as

$$S(f) = \{\alpha \in \mathbb{Z}^n_{\geq 0} : c_\alpha \neq 0\}.$$

A set of n polynomials $\{f_1, \ldots, f_n\} \subset \mathbb{C}[x]$ defines a *square ideal*

$$I = \langle f_1, \ldots, f_n \rangle = \{g_1 f_1 + \ldots + g_n f_n : g_1, \ldots, g_n \in \mathbb{C}[x]\} \subset \mathbb{C}[x].$$

The *affine variety* associated to I is

$$\mathbb{V}(I) = \{x \in \mathbb{C}^n : f(x) = 0, \forall f \in I\} = \{x \in \mathbb{C}^n : f_1(x) = \ldots = f_n(x) = 0\}.$$

In this paper, we assume that the variety $\mathbb{V}(I)$ consists of finitely many points $\{z_1, \ldots, z_N\} \subset \mathbb{C}^n$. Such a variety is called *0-dimensional*.

*Supported by the Research Council KU Leuven, PF/10/002 (Optimization in Engineering Center (OPTEC)), C1-project (Numerical Linear Algebra and Polynomial Computations), by the Fund for Scientific Research–Flanders (Belgium), G.0828.14N (Multivariate polynomial and rational interpolation and approximation), and by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization).

A well known result in algebraic geometry states that the quotient ring $k[x_1, \ldots, x_n]/I$ with $I \subset k[x_1, \ldots, x_n]$ a 0-dimensional ideal and k an algebraically closed field is isomorphic as a k-algebra to a finite dimensional k-vectorspace V with multiplication defined by a pairwise commuting set of n square matrices over k. This set of matrices corresponds to a set of generators of $k[x_1, \ldots, x_n]/I$ and the size of each matrix is equal to the number of points in $\mathbb{V}(I) \subset k^n$, counting multiplicities. Once the (generating) multiplication matrices are known in some basis, we can answer several questions about the variety $\mathbb{V}(I)$. For example, we can retrieve the solutions of the system by computing their eigenstructure and we can evaluate any polynomial on $\mathbb{V}(I)$. Our goal is to compute the multiplication matrices in a numerically stable way for square ideals satisfying some genericity assumptions.

There are many approaches to the problem of solving systems of polynomial equations. The different methods are often subdivided in homotopy methods, subdivision methods and algebraic methods. Homotopy continuation uses Newton iteration to track solution paths, starting from a simple initial system and gradually transforming it into the target system. These ideas have led to highly successful solvers [1, 24]. However, performing some numerical experiments one observes that for large systems some solutions might be lost along the way. The continuation gives up on certain paths when, for example, they seem to be diverging to infinity or they enter an ill-conditioned region. Normal form algorithms belong to the category of algebraic methods. The earliest versions of these algorithms use Groebner bases [6, 7] and doing so they make an implicit choice of basis for $\mathbb{C}[x]/I$. It turns out that these methods are numerically unstable and infeasible for large systems of equations (high degree, many variables). More recent algorithms are based on *border bases* [19, 23, 20]. Essentially, they fix a basis $\mathcal{O}$ for $\mathbb{C}[x]/I$ and construct the multiplication matrices of the coordinate functions by calculating the normal forms of $x_1 \cdot \mathcal{O}$, $\ldots$, $x_n \cdot \mathcal{O}$ with respect to $\mathcal{O}$. Border bases are a generalization of Groebner bases and they can be used to enhance the numerical stability of normal form algorithms. However, there are no algorithms that make a choice of $\mathcal{O}$ based on the conditioning of the normal form computation problem. This is mentioned as an open problem in [20]. In this paper we present such an algorithm for generic systems that makes an automatic choice of $\mathcal{O}$, which does not necessarily correspond to a Groebner basis, nor to a border basis. What is meant by 'generic systems' is explained in Section 2. The goal is to cover the generic, dense case to illustrate the effectiveness of the idea. The connection with resultant algorithms for dense systems is established. This suggests that the techniques can be generalized to sparse systems of equations. Such a generalization will follow from the sparse variant of the Macaulay resultant algorithm, see for instance [12].

In the following section we discuss our genericity assumptions and some properties of the systems that satisfy them. Section 3 briefly reviews the multiplication maps in $\mathbb{C}[x]/I$ and their properties. We give a short motivation in Section 4 by discussing some aspects of Macaulay's resultant construction and border bases algorithms that are generalized in our approach. In Section 5 we introduce a construction that we call *Macaulay matrices*. Section 6 presents the algorithm and some connections with border bases and Macaulay resultants. In the final section we present some numerical experiments.

2 Generic total degree systems

We say that a polynomial $f \in \mathbb{C}[x] \backslash \{0\}$ is of degree d if

$$\max_{\alpha \in S(f)} |\alpha| = d,$$

where $|\alpha| = \alpha_1 + \ldots + \alpha_n$. We denote $\deg(f) = d$. Accordingly, we say that a square polynomial system in n variables given by $\{f_1, \ldots, f_n\}$ is of degree $(d_1, \ldots, d_n)$ if $\deg(f_i) = d_i, i = 1, \ldots, n$. A polynomial $f \in \mathbb{C}[x]\backslash\{0\}$ is called *homogeneous* of degree d if $|\alpha| = d, \forall \alpha \in S(f)$.
Consider the *projective n-space*

$$\mathbb{P}^n = (\mathbb{C}^{n+1}\backslash\{0\})/\sim,$$

where $(a_0, \ldots, a_n) \sim (b_0, \ldots, b_n)$ iff $a_i = \lambda b_i, i = 0, \ldots, n, \lambda \in \mathbb{C}\backslash\{0\}$. We can interpret $\mathbb{P}^n$ as the union of $n+1$ copies of $\mathbb{C}^n$, each of them given by putting one of the coordinates equal to 1. We will also think of $\mathbb{P}^n$ as the union of $\mathbb{C}^n$ corresponding to $x_0 = 1$ and the set $\{x_0 = 0\}$, called the *hyperplane at infinity*. For more on projective space, see [6]. Note that the equation $f = 0$ with $f \in \mathbb{C}[x_0, \ldots, x_n]$ is well defined over $\mathbb{P}^n$ if and only if f is homogeneous. Starting from a polynomial $f \in \mathbb{C}[x]$ in n variables of degree d, we can obtain a homogeneous polynomial $f^h \in \mathbb{C}[x_0, \ldots, x_n]$, called the *homogenization* of f as

$$f^h = x_0^d f\left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\right).$$

The following theorem was proved by Étienne Bézout for the intersection of algebraic plane curves in $\mathbb{P}^2$. The generalization is often referred to as Bézout's theorem.

Theorem 1 (Bézout)**.** *A system of n homogeneous equations of degree $(d_1, \ldots, d_n)$ in $n+1$ variables with a finite number of solutions in $\mathbb{P}^n$ has exactly $d_1 \cdots d_n$ solutions in $\mathbb{P}^n$, counting multiplicities.*

Proof. The theorem is a corollary of Theorem 7.7 in [16]. □

It is not difficult to show that for *almost all* systems with degree $(d_1, \ldots, d_n)$, all $d_1 \cdots d_n$ solutions lie in the overlapping part of the affine charts of $\mathbb{P}^n$ [7]. Hence, if the n homogeneous equations in $n+1$ variables of Theorem 1 are the homogenizations of n affine equations $f_1 = \ldots = f_n = 0$ in n variables, all of the $d_1 \cdots d_n$ solutions correspond to points in $\mathbb{C}^n \subset \mathbb{P}^n$.

The kind of systems that we consider in this paper are the ones that satisfy the assumption of Bézout's theorem. Namely, we assume that the homogenized equations $f_1^h = \ldots = f_n^h = 0$ have a finite number of solutions in $\mathbb{P}^n$. We denote $\overline{I} = \langle f_1^h, \ldots, f_n^h \rangle$ and $\mathbb{V}(\overline{I}) = \{x \in \mathbb{P}^n : f_1^h(x) = \ldots = f_n^h(x) = 0\}$. Furthermore, we assume that none of the solutions lie on the hyperplane at infinity. Note that this last assumption is not really restrictive: a random linear change of projective coordinates will move all of the solutions away from the hyperplane $\{x_0 = 0\}$ with probability 1.

3 Multiplication in $\mathbb{C}[x]/I$

In this section we briefly review the $\mathbb{C}$-algebra structure of the quotient ring $\mathbb{C}[x]/I$ and the properties of multiplication in this ring. For an extensive treatment one can consult [6, 7, 23, 11]. Consider the following equivalence relation on $\mathbb{C}[x]$:

$$f \sim g \Leftrightarrow f - g \in I.$$

Now, every polynomial $f \in \mathbb{C}[x]$ defines a residue class $[f] = f + I$ with respect to $\sim$. We call the polynomial f a *representative* of the residue class $[f]$. The set of all such residue classes is the *quotient ring* $\mathbb{C}[x]/I$. Note that $[0] = I$. One can check that the scalar multiplication and addition operations

$$\alpha[f] = [\alpha f], \qquad [f] + [g] = [f + g] \tag{1}$$

with $\alpha \in \mathbb{C}$ and $f, g \in \mathbb{C}[x]$ are well defined. This implies that $\mathbb{C}[x]/I$ is a vector space. Moreover, to show that $\mathbb{C}[x]/I$ is a $\mathbb{C}$-algebra, it can be checked that multiplication

$$[f] \cdot [g] = [fg]$$

is well defined. The following theorem allows us to describe these operations on $\mathbb{C}[x]/I$ using linear algebra.

Theorem 2. *For a zero-dimensional ideal I, the dimension of $\mathbb{C}[x]/I$ as a vector space is equal to the number of points in $\mathbb{V}(I) \subset \mathbb{C}^n$, counting multiplicities.*

Proof. For the proof of this theorem we refer to [7]. □

We now consider the map $m_f : \mathbb{C}[x]/I \to \mathbb{C}[x]/I$ given by

$$m_f([g]) = [f] \cdot [g] = [fg], \forall g \in \mathbb{C}[x].$$

This map is linear, so once we choose a basis $\mathcal{O}$ for $\mathbb{C}[x]/I$, it can be represented by an $N \times N$ matrix, where N is the number of solutions (counting multiplicities). Under our genericity assumptions, N is the Bézout number: $N = \prod_{i=1}^{n} d_i$. Once we have fixed a basis of $\mathbb{C}[x]/I$, we will no longer make a distinction between the map m_f and its matrix representation. The matrix representing multiplication by f is called a *multiplication matrix* of f. Its eigenstructure has the following remarkable properties.

Theorem 3. *Let I be a zero-dimensional ideal in $\mathbb{C}[x]$ and let m_f be the multiplication matrix of $f \in \mathbb{C}[x]$ with respect to a given basis $\mathcal{O} = \{[b_1], \ldots, [b_N]\}$ of $\mathbb{C}[x]/I$. Then*

$$\det(m_f - \lambda \mathbb{I}) = (-1)^N \prod_{z \in \mathbb{V}(I)} (\lambda - f(z))^{\mu(z)}$$

where $N = \dim \mathbb{C}[x]/I$, $\mathbb{I}$ is the identity matrix of size $N \times N$ and $\mu(z)$ is the multiplicity of the root z. Also, the row vector

$$\begin{bmatrix} b_1(z) & \cdots & b_N(z) \end{bmatrix}$$

lies in the left eigenspace of the eigenvalue $f(z)$ for each $z \in \mathbb{V}(I)$[1].

Proof. For the proof, we refer the reader to [7, Chapter 4]. □

Theorem 3 implies that if we want to compute the coordinates of the solutions $z_1, \ldots, z_N$, we can construct the multiplication matrices $m_{x_1}, \ldots, m_{x_N}$ corresponding to the coordinate functions and compute their eigenvalues. Another possibility is to use the eigenvectors [23, 7]. Note that, according to Theorem 3, the left eigenvectors do not depend on the choice of f. In fact, neither do the right ones. By their definition, it is not difficult to see that the multiplication maps must commute. They form a family of commuting matrices, so they must share common eigenspaces [23]. We note here that when the set of eigenvectors spans $\mathbb{C}^N$ (that is, when all solutions are simple), the matrices $m_{x_1}, \ldots, m_{x_n}$ are simultaneously diagonalizable. We will give an example of the construction of the multiplication matrices of the coordinate functions in Section 6. To work out this example, we will need the notion of a *normal form*.

Definition 1 (Normal form)**.** *Let $\mathcal{O} = \{[b_1], \ldots, [b_N]\}$ be a basis for $\mathbb{C}[x]/I$. Given any polynomial $g \in \mathbb{C}[x]$, let*

$$[g] = a_1[b_1] + \ldots + a_N[b_N] = [a_1 b_1 + \ldots + a_N b_N], \quad a_i \in \mathbb{C}$$

be the unique representation of $[g]$ in the basis $\mathcal{O}$. We say that $a_1 b_1 + \ldots + a_N b_N$ is the normal form *of g w.r.t. $\mathcal{O}$. We denote $\overline{g}^{\mathcal{O}} = a_1 b_1 + \ldots + a_N b_N$.*

[1] Note that in general $\#\mathbb{V}(I) \leq N$ where equality only holds if all solutions are simple.

Note that for any $g \in \mathbb{C}[x]$, if the basis elements $[b_i]$ are given by monomials: $[b_i] = [x^{\alpha_i}]$, we have that $S(\overline{g}^{\mathcal{O}}) \subset \{\alpha_1, \ldots, \alpha_N\}$. In general $S(\overline{g}^{\mathcal{O}}) \subset \bigcup_{i=1}^{N} S(b_i)$. The results in this section show that normal form algorithms for generic systems can be divided into two major parts.

1. Compute the multiplication operators m_{x_i}, $i = 1, \ldots, n$.
2. Perform a simultaneous diagonalization of the m_{x_i} to find the solutions or find the solutions via the eigenvectors of the m_{x_i}.

In this paper, we focus on making improvements in the first step. The proposed algorithm will choose a basis $\mathcal{O}$ for $\mathbb{C}[x]/I$ such that the multiplication operators can be computed, heuristically, as accurately as possible.

4 Motivation

The normal form method presented in this paper is closely related to border basis algorithms and to multipolynomial Macaulay resultants. We briefly review some of their properties that are exploited or generalized in our algorithm. For more details on border bases we refer to [19, 11], and for multipolynomial resultants to [7].

4.1 Macaulay resultant matrices

Consider the system of homogenized equations $f_1^h = \ldots = f_n^h = 0$ coming from $I = \langle f_1, \ldots, f_n \rangle$. As discussed in Section 2, the expected number of solutions in $\mathbb{P}^n$ is finite and equal to Bézout's number. If we add a generic homogeneous equation $f_0^h = 0$ to the system, then generically the system has no solutions. The *resultant* $\mathrm{Res}(f_0^h, \ldots, f_n^h)$ is a homogeneous polynomial in the coefficients of the f_i^h that vanishes if and only if the system $f_0^h = \ldots = f_n^h = 0$ has a solution in $\mathbb{P}^n$. A resultant matrix M_0 is a matrix such that $\det(M_0)$ is a nonzero multiple of the resultant polynomial. Several constructions of resultant matrices have been introduced [11, 7, 5]. The one that is related in the most direct way to the algorithm presented in this paper is a generalization of the Sylvester matrix of two univariate polynomials to the multivariate case, also called the multipolynomial Macaulay resultant matrix [7]. The rows in this matrix correspond to monomial multiples of the input equations, whereas its columns correspond to monomials, such that the coefficient of the polynomial corresponding to the j-th row coupled to the i-th monomial is the (j, i) entry of the matrix. We denote this matrix by M_0. Let f_0^h be a generic linear form and let M_0 be the Macaulay resultant matrix associated to $f_0^h, f_1^h, \ldots, f_n^h$. We view it as a block matrix

$$M_0 = \begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix},$$

such that the first block row $[M_{00}\ M_{01}]$ contains the multiples of f_0^h by the set of monomials

$$\mathcal{O}_M = \{x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} : 0 \leq \alpha_i \leq d_i - 1, 1 \leq i \leq n\} \tag{2}$$

and the second block row contains the monomial multiples of the other f_i^h. Furthermore, we assume that the columns are ordered in such a way that the first block column $\begin{bmatrix} M_{00} \\ M_{10} \end{bmatrix}$ corresponds to the monomials in $\mathcal{O}_M$. In [7, Chapter 3] it is shown that, with our genericity assumptions, we have that the Schur complement $m_{f_0} = M_{00} - M_{10} M_{11}^{-1} M_{01}$ represents multiplication by $f_0 = f_0^h(1, x_1, \ldots, x_n)$ in $\mathbb{C}[x]/I$ in the basis $\mathcal{O}_M$. Macaulay [18] showed that generically, M_{11} is invertible and hence it is possible to compute this Schur complement. One could, for instance, use $f_0^h = x_i$ to find m_{x_i} in this way and find the solutions of I by using the results in Section 3.

This leads to a well known eigenvalue-eigenvector method for solving generic dense systems, based on resultants. However, when computations are performed in finite precision, the accuracy of the resulting matrix m_{f_0} depends on the condition number of the inversion of M_{11}. That is, the 'more invertible' M_{11} is, the more accurate the operator m_{f_0} can be obtained from this matrix, hence the more accurate we can compute its eigenstructure to find the points defined by I. The algorithm proposed in this paper somehow chooses the partitioning of M_0 in an adaptive way, such that M_{11} is well conditioned and the Schur complement still gives the multiplication map.

4.2 Border bases

Groebner bases and Buchberger's algorithm to compute them provide an algorithmic, algebraic way to compute the solutions of a system of polynomial equations [6, 4, 13]. They can be used to compute normal forms in a basis for $\mathbb{C}[x]/I$ induced by a monomial order. A major drawback is that for large problems, Groebner bases are not feasible in finite precision, since the computations are unstable. Border bases have been developed as a generalization of Groebner bases to represent the quotient algebra $\mathbb{C}[x]/I$ [21, 22, 19]. With respect to Groebner bases, they enhance the numerical stability due to a more flexible choice of monomial bases for $\mathbb{C}[x]/I$ and they are also more robust (the border basis remains a basis for small perturbations of the coefficients) [20]. A border basis $\mathcal{O}$ has the property that it is *connected to 1*. This means that $1 \in \text{span}(\mathcal{O})$ and and, for any $g \in \text{span}(\mathcal{O})$ there are $g_1, \ldots, g_n \in \text{span}(\mathcal{O})$ such that

$$g = \sum_{i=1}^{n} x_i g_i.$$

The border basis criterion for normal form algorithms is given by the following theorem [19].

Theorem 4. *Let $B = \text{span}(\mathcal{O}) \subset \mathbb{C}[x]$ be such that $\mathcal{O}$ is connected to 1. Let $N : B \cup (\bigcup_{i=1}^{n} x_i \cdot B) \to B$ be a $\mathbb{C}$-linear map such that it is the identity restricted to B. Let $I = \langle \ker N \rangle$ be the ideal generated by the kernel of N. Defining $M_i : B \to B : b \mapsto N(x_i b)$, the following properties are equivalent:*

1. *$M_i \circ M_j = M_j \circ M_i$,*
2. *$\mathbb{C}[x] = B \oplus I$.*

From $\mathbb{C}[x] = \mathbb{C}[x]/I \oplus I$ it follows that when the M_i from Theorem 4 commute, $B \simeq \mathbb{C}[x]/I$ as $\mathbb{C}$-algebras and M_i represents multiplication with x_i modulo I since $I = \langle \ker(N) \rangle$. Therefore, a basis $\mathcal{O}$ must not be induced by a monomial order. It is sufficient that $\mathcal{O}$ is connected to 1 and there is a map N with the right properties: its kernel generates I and the maps M_i are pairwise commuting. Note that the basis $\mathcal{O}_M$ from (2) is connected to 1. We will show in Section 7 that this basis can still show bad numerical behaviour. In this paper we propose an algorithmic choice of basis that does not necessarily have the connected to 1 property. In border basis algorithms, the basis $\mathcal{O}$ is fixed beforehand. This has the advantage that the algorithm can be adapted to this specific basis to reduce the computational cost. However, the choice of basis can influence the accuracy dramatically, as we will show in Section 7. As specified in the following sections, the algorithm presented in this paper starts from a set of candidate monomials (which we will denote later by $S(M)_{<t}$) from which we will select the monomials in $\mathcal{O}$. This set is quite large, and to choose the basis $\mathcal{O}$ the algorithm uses numerical linear algebra techniques on a matrix called the *Macaulay matrix*, very similar to Macaulay's resultant construction.

5 Macaulay matrices

A *Macaulay matrix* associated to the set of polynomials $\{f_1, \ldots, f_n\} \subset \mathbb{C}[x]$ is a matrix over $\mathbb{C}$ in which each column corresponds to a monomial $x^\alpha, \alpha \in \mathbb{Z}^n_{\geq 0}$. Furthermore, such a Macaulay matrix has n block rows, each of which corresponds to one of the polynomials in the set. The j-th row of the i-th block row is the vector representation of a polynomial $x^{\beta_{ij}} f_i \in I, \beta_{ij} \in \mathbb{Z}^n_{\geq 0}$ in the basis $\{x^\alpha\}$ of monomials corresponding to the columns. For example, denote $R = \mathbb{C}[x]$ and for an ideal $J \subset R$, we denote by $J_{\leq t}$ the elements in J of degree $\leq t$. Let $d_i = \deg(f_i)$. For $t \geq \max_i d_i$, consider the linear map

$$\begin{aligned} &\bigoplus_{i=1}^{n} R_{\leq t-d_i} \longrightarrow I_{\leq t}, \\ &(a_1, \ldots, a_n) \longrightarrow a_1 f_1 + \cdots + a_n f_n. \end{aligned}$$

The transpose of the matrix representation of this map with respect to the standard monomial basis of $R_{\leq t}$ is a Macaulay matrix. We will call such a Macaulay matrix a *dense Macaulay matrix*. We clarify this by means of an example.

Example 1. *Let $I = \langle f_1, f_2 \rangle \subset \mathbb{C}[x_1, x_2]$ be generated by $f_1 = a + bx_1 + cx_2$ and $f_2 = d + ex_1 + fx_2 + gx_1^2 + hx_1x_2 + jx_2^2$ with $a, \ldots, j \in \mathbb{C}$. It is clear that $I_{\leq 2}$ is a subset of $R_{\leq 2} = \mathbb{C}[x_1, x_2]_{\leq 2}$ which is spanned as a $\mathbb{C}$-vector space by $1, x_1, x_2, x_1^2, x_1x_2, x_2^2$. Using this basis to represent elements of $I_{\leq 2}$ and $R_{\leq 1} \oplus R_{\leq 0} = \operatorname{span}(1, x_1, x_2) \oplus \operatorname{span}(1)$ we get the transpose of the matrix*

$$M = \begin{array}{c} \\ f_1 \\ x_1 f_1 \\ x_2 f_1 \\ f_2 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{array} \\ \begin{bmatrix} a & b & c & & & \\ & a & & b & c & \\ & & a & & b & c \\ d & e & f & g & h & j \end{bmatrix} \end{array}$$

for the matrix representation of

$$\begin{aligned} &R_{\leq 1} \oplus R_{\leq 0} \longrightarrow I_{\leq 2}, \\ &(a_1, a_2) \longrightarrow a_1 f_1 + a_2 f_2. \end{aligned}$$

The matrix M is clearly a Macaulay matrix.

By the support of M, we mean the set of exponent vectors

$$S(M) = \{\alpha \in \mathbb{Z}^n_{\geq 0} : x^\alpha \text{ corresponds to a column of } M\}.$$

To describe the row content of M, we define the sets

$$\Sigma_i(M) = \{\beta_{ij} \in \mathbb{Z}^n_{\geq 0} : x^{\beta_{ij}} f_i \text{ corresponds to a row of the } i\text{-th block row of } M\}.$$

The set Σ_i is also called the set of shifts of f_i. Note that, given the polynomials f_i, M is defined up to row and column permutations by $S(M)$ and $\Sigma_i(M)$, $1 \leq i \leq n$ and in order to be feasible, these sets must satisfy

$$S(x^{\beta_{ij}} f_i) \subset S(M), \forall \beta_{ij} \in \Sigma_i(M), 1 \leq i \leq n.$$

Example 2. *In the previous example, we have $S(M) = \{(0,0), (1,0), (0,1), (2,0), (1,1), (0,2)\}$, $\Sigma_1(M) = \{(0,0), (1,0), (0,1)\}$, $\Sigma_2(M) = \{(0,0)\}$.*

A Macaulay matrix of this type has a natural homogeneous interpretation. We show this by continuing the previous example.

Example 3. *Homogenizing the equations we get* $f_1^h = ax_0 + bx_1 + cx_2$ *and* $f_2^h = dx_0^2 + ex_0x_1 + fx_0x_2 + gx_1^2 + hx_1x_2 + jx_2^2$*, where the superscript* h *indicates the homogenization and it should not be confused with the coefficient* $h \in \mathbb{C}$ *of the monomial* x_1x_2 *in* f_2*. We denote* $\overline{I} = \langle f_1^h, f_2^h \rangle \subset \mathbb{C}[x_0, x_1, x_2]$*. Now, one can verify that* M *is also the Macaulay matrix of* $\{f_1^h, f_2^h\}$ *with*

$$S^h(M) = \{(2,0,0), (1,1,0), (1,0,1), (0,2,0), (0,1,1), (0,0,2)\} \subset \mathbb{Z}^3,$$

$\Sigma_1^h(M) = \{(1,0,0), (0,1,0), (0,0,1)\}$, $\Sigma_2^h(M) = \{(0,0,0)\}$*. It is clear how this can be generalized to any dense Macaulay matrix: if the associated map has image in* $I_{\leq t}$*, homogenize the exponent vectors in* $S(M)$ *to degree* t *in* $\mathbb{Z}_{\geq 0}^{n+1}$ *and the exponent vectors in* $\Sigma_i(M)$ *to degree* $t - d_i$*. The associated linear map is given as follows. Denoting* $R^h = \mathbb{C}[x_0, x_1, \ldots, x_n]$ *and the degree* t *part of a graded* R^h*-module* A *by* A_t *(the grading is induced by the standard grading on* R^h*),* M *represents the map*

$$\begin{aligned} &\bigoplus_{i=1}^{n} R_{t-d_i}^h \longrightarrow \overline{I}_t, \\ &(a_1, \ldots, a_n) \longrightarrow a_1 f_1^h + \cdots + a_n f_n^h. \end{aligned}$$

This map is surjective.

Macaulay matrices are used to give determinantal formulations of resultants [7] and to solve systems of polynomial equations [7, 9, 3]. They form a natural first step in reformulating the root finding problem as a linear algebra problem. The following theorem is straightforward [9].

Theorem 5. *Let* $S(M) = \{\alpha_1, \ldots, \alpha_l\}$ *be the support of a Macaulay matrix* M *of* $\{f_1, \ldots, f_n\}$*, where* α_i *corresponds to the* i*-th column of* M*. Let* $I = \langle f_1, \ldots, f_n \rangle$*. The point* $z \in \mathbb{C}^n$ *satisfies* $z \in \mathbb{V}(I)$ *if and only if the vector*

$$v(z) = (x^{\alpha_1}(z), \ldots, x^{\alpha_l}(z))^\top$$

satisfies $Mv(z) = 0$*.*

It is clear that Theorem 5 generalizes to the projective interpretation of M.

Theorem 6. *Let* $S^h(M) = \{\alpha_1^h, \ldots, \alpha_l^h\}$ *be the support of a (homogeneously interpreted) Macaulay matrix* M *of* $\{f_1^h, \ldots, f_n^h\}$*, where* α_i^h *corresponds to the* i*-th column of* M*. Denote* $\overline{I} = \langle f_1^h, \ldots, f_n^h \rangle$*. The point* $z^h \in \mathbb{P}^n$ *satisfies* $z^h \in \mathbb{V}(\overline{I})$ *if and only if the point*

$$v(z^h) = (x^{\alpha_1^h}(z^h), \ldots, x^{\alpha_l^h}(z^h))^\top,$$

viewed as a point in $\mathbb{P}^{l-1}$*, satisfies* $Mv(z^h) = 0$ *(note that here* $x = (x_0, x_1, \ldots, x_n)$ *is short for an* $n+1$*-tuple). This condition is well defined, since* $v(\lambda z^h) = \lambda^t v(z^h), \lambda \in \mathbb{C}\backslash\{0\}$ *and* $t = |\alpha_i^h|$*.*

Theorem 6 implies that every point $z^h \in \mathbb{V}(\overline{I}) \subset \mathbb{P}^n$ generates a direction $v(z^h)$ in the nullspace of M. We will now present a way to construct the dense Macaulay matrix such that its null space is spanned by these directions. In the Macaulay matrix M with support $S(M) = \{\alpha \in \mathbb{Z}_{\geq 0}^n : |\alpha| \leq t\}$, the number of columns is

$$\#S(M) = \binom{t+n}{n}.$$

Consider the shifts

$$\Sigma_i(M) = \{\beta \in \mathbb{Z}_{\geq 0}^n : |\beta| \leq t - d_i\}.$$

It is clear that the resulting matrix M is the dense Macaulay matrix of degree t.

Theorem 7. *Under our genericity assumptions, for M constructed as above with $t \geq \sum_{i=1}^{n} d_i - n$, we have* $\dim \operatorname{null}(M) = N$. *Equivalently, for these values of t:* $\#S(M) - N = \operatorname{rank}(M)$.

Proof. This result was known by Macaulay [18]. The degree $t = \sum_{i=1}^{n} d_i - n$ is called the *degree of regularity* in [3, 9]. □

In fact, we have by construction that for the Macaulay matrix of degree t, $\dim \operatorname{null}(M)$ is the codimension of $\overline{I}_t$ in R_t^h, which is the dimension of $(R^h/\overline{I})_t$ as a $\mathbb{C}$-vector space. This is the Hilbert function of $\overline{I}$ evaluated at t [7, 10]. Since $\overline{I}$ defines points in $\mathbb{P}^n$ by assumption, the Hilbert function becomes constant for large t and Theorem 7 implies that this happens at $t = \sum_{i=1}^{n} d_i - n$. In the algorithm, we will also rely on the following theorem.

Theorem 8. *For $t \geq \sum_{i=1}^{n} d_i - (n-1)$ we have that*

$$\binom{t-1+n}{n} \geq N,$$

with $N = \prod_{i=1}^{n} d_i$.

Proof. The number $\binom{t-1+n}{n}$ is the number of monomials of degree at most $t-1 = \sum_{i=1}^{n} d_i - n$. The number $N = \prod_{i=1}^{n} d_i$ is the number of monomials in the set $\{\alpha \in \mathbb{Z}_{\geq 0}^n : \alpha_i \leq d_i - 1, i = 1, \ldots, n\}$. The highest degree monomial in this set has degree $\sum_{i=1}^{n} d_i - n$. □

It will become clear later that the properties of M given in Theorem 7 and Theorem 8 are exactly the properties we need in our algorithm. We also want M to be as small as possible to reduce memory use and computational effort. We therefore set $t = \sum_{i=1}^{n} d_i - (n-1)$.

6 Normal form computation using the Macaulay matrix

In this section, we propose a new normal form algorithm for computing the m_{x_i} for a generic dense system as described in Section 2.

6.1 An example

We introduce the ideas of our algorithm by a simple example. Consider the ideal $I = \langle f_1, f_2 \rangle \subset \mathbb{C}[x_1, x_2]$ given by $f_1(x_1, x_2) = x_1^2 + x_2^2 - 2 = 0, f_2(x_1, x_2) = 3x_1^2 - x_2^2 - 2 = 0$. We will use linear combinations of $f_1, xf_1, yf_1, f_2, xf_2, yf_2$ to find the normal forms. The variety $\mathbb{V}(I) = \{(-1,-1), (-1,1), (1,-1), (1,1)\}$ is 0-dimensional and the system satisfies the genericity assumptions. A possible basis for $\mathbb{C}[x_1, x_2]/I$ is $\mathcal{O} = \{[1], [x_1], [x_2], [x_1 x_2]\}$. We construct the dense Macaulay matrix M of degree $t = \sum_{i=1}^{2} d_i - (n-1) = 3$ as presented in Section 5, ordering the columns such that these monomials correspond to the last four columns:

$$M = \begin{array}{c} \\ f_1 \\ x_1 f_1 \\ x_2 f_1 \\ f_2 \\ x_1 f_2 \\ x_2 f_2 \end{array} \begin{array}{c} \begin{array}{cccccc|cccc} x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 & x_1^2 & x_2^2 & \underline{1} & \underline{x_1} & \underline{x_2} & \underline{x_1 x_2} \end{array} \\ \left[\begin{array}{cccccc|cccc} & & & & 1 & 1 & -2 & & & \\ 1 & & 1 & & & & & -2 & & \\ & 1 & & 1 & & & & & -2 & \\ & & & & 3 & -1 & -2 & & & \\ 3 & & -1 & & & & & -2 & & \\ & 3 & & -1 & & & & & -2 & \end{array}\right] \end{array}.$$

To construct the multiplication maps m_{x_1} and m_{x_2} with respect to $\mathcal{O}$, we need to calculate the normal forms of $x_1^2, x_1^2 x_2, x_2^2, x_1 x_2^2$ in $\mathcal{O}$. All of these monomials appear in the left block column of M. Inverting this column block and applying it from the left to M gives

$$\tilde{M} = \begin{array}{c} \\ x_1^3 - x_1 \\ x_1^2 x_2 - x_2 \\ x_1 x_2^2 - x_1 \\ x_2^3 - x_2 \\ x_1^2 - 1 \\ x_2^2 - 1 \end{array} \begin{array}{c} \begin{array}{cccccc|cccc} x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 & x_1^2 & x_2^2 & \underline{1} & \underline{x_1} & \underline{x_2} & \underline{x_1x_2} \end{array} \\ \left[\begin{array}{cccccc|cccc} 1 & & & & & & & -1 & & \\ & 1 & & & & & & & -1 & \\ & & 1 & & & & & -1 & & \\ & & & 1 & & & & & -1 & \\ & & & & 1 & & -1 & & & \\ & & & & & 1 & -1 & & & \end{array}\right] \end{array}.$$

Note that the left block was square because of the properties of the dense Macaulay matrix. The rows of $\tilde{M}$ are linear combinations of the rows of M, so they represent polynomials in I. Hence, for example, $[x_1^2 - 1] = [0]$ modulo I and the normal form of x_1^2 is 1. Using the information in $\tilde{M}$ we can construct m_{x_1} and m_{x_2}. This gives

$$m_{x_1} = \begin{array}{c} \\ [1] \\ [x_1] \\ [x_2] \\ [x_1x_2] \end{array} \begin{array}{c} \begin{array}{cccc} [x_1]\cdot[1] & [x_1]\cdot[x_1] & [x_1]\cdot[x_2] & [x_1]\cdot[x_1x_2] \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right] \end{array},$$

$$m_{x_2} = \begin{array}{c} \\ [1] \\ [x_1] \\ [x_2] \\ [x_1x_2] \end{array} \begin{array}{c} \begin{array}{cccc} [x_2]\cdot[1] & [x_2]\cdot[x_1] & [x_2]\cdot[x_2] & [x_2]\cdot[x_1x_2] \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right] \end{array}.$$

Note that the first and the third column of m_{x_1} are trivial and so are the first and the second column of m_{x_2}. The other columns can be read off $\tilde{M}$ directly. The eigenvalues of m_{x_i} coincide with the i-th coordinates of the points in $\mathbb{V}(I)$.

6.2 The monomial basis

When choosing the basis $\mathcal{O}$, we must take into account that $\mathcal{O}$ cannot contain monomials of degree t (3 in the previous example). Otherwise, multiplying with x_1 or x_2 gives a monomial that is not in $S(M)$. Secondly, it must be such that the resulting system is solvable. In the generic case, there is always such a choice. We consider the Macaulay matrix M of degree $t = \sum_{i=1}^n d_i - (n-1)$. By $S(M)_t$ we denote the monomials in $S(M)$ of degree $t = \sum_{i=1}^n d_i - n + 1$ and by $S(M)_{<t}$ the remaining monomials. We order the columns of the Macaulay matrix in such a way that

$$M = \begin{bmatrix} M_b & M_i & B \end{bmatrix}$$

where M_b are the columns corresponding to $S(M)_t$, B contains the columns corresponding to $\mathcal{O}$ and M_i corresponds to $S(M)_{<t} \backslash \mathcal{O}$. When the polynomials $f_1, \dots, f_n$ are generic, the set of monomials

$$\mathcal{O}_M = \{x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} : 0 \le \alpha_i \le d_i - 1, 1 \le i \le n\} \subset S(M)_{\le t} \quad (3)$$

is a basis for $\mathbb{C}[x]/I$ [7] (this is exactly the basis used in the Macaulay resultant construction, as introduced in Section 4). This means that every monomial in $S(M) \backslash \mathcal{O}_M$ has a unique normal form in $\mathcal{O}_M$. In other words, there is a unique polynomial of the form

$$g_\alpha = x^\alpha - \sum_{b \in \mathcal{O}_M} c_{\alpha b} b \in I \quad (4)$$

with $c_{\alpha b} \in \mathbb{C}$, for each $\alpha \in S(M)\backslash\mathcal{O}_M$. Also, it follows from Property (iii) in [5, Chapter 1, p.46] and from our assumptions that for $t \geq \sum_{i=1}^{n} d_i - (n-1)$, the rows of M span $I_{\leq t}$ linearly. Now, since $g_\alpha \in I_{\leq t}$ and the rows of M span $I_{\leq t}$, every g_α is a linear combination of the rows of M. The number of polynomials g_α is $r = \operatorname{rank}(M)$, so we can apply a square matrix to the left of M to transform M into

$$\left[\begin{array}{ccccccc} 1 & & & & -c_{\alpha_1 b_1} & \cdots & -c_{\alpha_1 b_N} \\ & 1 & & & -c_{\alpha_2 b_1} & \cdots & -c_{\alpha_2 b_N} \\ & & \ddots & & \vdots & & \vdots \\ & & & 1 & -c_{\alpha_r b_1} & \cdots & -c_{\alpha_r b_N} \\ \hline 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right]. \tag{5}$$

The block row of zeros is introduced by the syzygies in the rows of M[2] [10]. This proves the following theorem.

Theorem 9. *The matrix M_b is of full column rank under our assumptions, and there is at least one possible choice of $\mathcal{O}$ for which $\begin{bmatrix} M_b & M_i \end{bmatrix}$ is of rank r.*

However, there are many more choices for $\mathcal{O}$ than the 'block basis' from (3). From a numerical point of view, it turns out this is crucial to find the normal forms with high accuracy. The idea is simple: we choose $\mathcal{O}$ in such a way that $\begin{bmatrix} M_b & M_i \end{bmatrix}$ is 'as invertible as possible', i.e., it has a small condition number.

6.3 Algorithm

We propose to make the choice of basis $\mathcal{O}$ by using a QR factorization with optimal column pivoting on (part of) the Macaulay matrix. This is a well known numerical linear algebra algorithm to compute an upper triangularization of a column permuted version of a matrix such that the diagonal elements are, heuristically, as large as possible (in absolute value). See for instance [14]. This leads to Algorithm 1 for the generic dense case. We briefly go through the different steps of the algorithm.

- Step 2 is obvious. In step 3, we re-arrange the columns of M such that M_b contains the columns corresponding to $S(M)_t$ and M_* contains all of the other columns. The order within the block columns is of no importance. We represented this in Algorithm 1 by a column permutation matrix P_c. At this point, we do not split M_* into M_i and B as before. The actual choice of basis is made in step 6.

- In step 4, we compute a QR factorization of the border block M_b, to make this block column upper triangular in step 5. The matrix $\hat{R}_b$ is the square upper triangular part of R_b, it is a nonsingular matrix. At this point, the lower block row represents polynomials in I of degree $\leq t-1$.

- Step 6 is essential. We perform a QR factorization *with optimal column pivoting* to the full lower right block. That is, we do not compute a QR factorization of $\hat{M}_*$, but of a column permuted version $\hat{M}_* P_i$, where P_i is a column permutation matrix. The column permutation is such that it heuristically selects the 'linearly most independent' columns first. In step 7 we apply the corresponding permutation to the entire matrix M and in step 8 we make the

[2] This occurs only for $n \geq 3$, not in the example given here.

Algorithm 1 Multiplication maps of a dense system

1: **procedure** MULTMATRICES($f_1, \ldots, f_n$)
2: $M \leftarrow$ dense Macaulay matrix of degree $\sum_{i=1}^{n} d_i - (n-1)$
3: $M \leftarrow \begin{bmatrix} M_b & M_* \end{bmatrix} = MP_c$
4: $M_b = Q_b R_b$
5: $M \leftarrow Q_b^* M = \begin{bmatrix} \hat{R}_b & Z \\ 0 & \hat{M}_* \end{bmatrix}$
6: $\hat{M}_* P_i = Q_i R_i$
7: $M \leftarrow M \begin{bmatrix} \mathbb{I} & 0 \\ 0 & P_i \end{bmatrix}$
8: $M \leftarrow \begin{bmatrix} \mathbb{I} & \\ & Q_i^* \end{bmatrix} M = \begin{bmatrix} \hat{R}_b & ZP_i \\ 0 & \tilde{R}_i \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \hat{R}_b & \hat{Z}_1 & \hat{Z}_2 \\ 0 & \hat{R}_i & \hat{Z}_3 \\ 0 & 0 & 0 \end{bmatrix}$
9: $M \leftarrow \begin{bmatrix} \hat{R}_b & \hat{Z}_1 & \hat{Z}_2 \\ 0 & \hat{R}_i & \hat{Z}_3 \end{bmatrix}$
10: $C \leftarrow -\begin{bmatrix} \hat{R}_b & \hat{Z}_1 \\ 0 & \hat{R}_i \end{bmatrix}^{-1} \begin{bmatrix} \hat{Z}_2 \\ \hat{Z}_3 \end{bmatrix}$
11: **for** $i = 1, \ldots, n$ **do**
12: Construct m_{x_i} using the normal forms in C.
13: **end for**
14: **return** $m_{x_1}, \ldots, m_{x_n}$
15: **end procedure**

entire matrix upper triangular ($\tilde{R}_i$ is the upper non-zero block and $\hat{R}_i$ is the square upper triangular part of $\tilde{R}_i$). We split the right block column into two block columns such that $\hat{R}_i$ is square. Under our assumptions, $\hat{R}_i$ is of full rank. Note that in the result, columns are still associated to monomials and the rows are polynomials in I. With increasing row index, the support of these polynomials is contained in a shrinking subset of $S(M)$. Note that in this step, the syzygies introduce a block row of zeros in M. We drop this block row of zeros in step 9. Denoting $r = \operatorname{rank}(M)$, the remaining matrix M is of size $r \times (N + r)$ by the results of Section 5.

- In step 10, we take out the leftmost $r \times r$ upper triangular block and apply its inverse to the right most $r \times N$ part with opposite sign to find the normal forms of all the monomials corresponding to the first r columns. Of course, we do not calculate the inverse, but apply backsubstitution instead. It is the condition number of this inversion that is controlled by the optimal column pivoting in step 6.

6.4 Connection with resultants and border bases

To show the relation with the Macaulay resultant construction we assume that $\mathcal{O}_M$ from (3) is a basis for $\mathbb{C}[x]/I$. Note that the row space of M is equal to the row space of the second block row of M_0, denoted by $[M_{10}\ M_{11}]$ in Section 4. It is isomorphic to the subvector space $\overline{I}_t$ (the degree t part) of $\overline{I}$. The resultant construction uses monomial multiples of the input equations that generically generate this subspace. Our construction uses all of the possible monomial multiples, which leads to more computational effort since we need to perform a row compression (step 8), but we observe numerically that the computed basis of the row space after this compression (first two block rows of M in step 8) has larger singular values.

Suppose that in steps 6, 7 we choose the basis $\mathcal{O}_M$ from (3) instead of performing the QR factorization with pivoting. We can apply an invertible transformation to M in step 9 on the left such that it becomes equal to $[M_{10}\ M_{11}]$ up to column permutation. In fact, by construction, M_{11} corresponds to the square, invertible, upper triangular part $\begin{bmatrix}\hat{R}_b & \hat{Z}_1\\ 0 & \hat{R}_i\end{bmatrix}$ of M, since the basis monomials correspond to M_{10} and $\begin{bmatrix}\hat{Z}_2\\ \hat{Z}_3\end{bmatrix}$. Choosing another basis $\mathcal{O}$ yields another matrix M_{11}. As long as it remains invertible and the multiples $\mathcal{O}\cdot f_0$ are supported in $S(M)$, m_{f_0} can be computed as the Schur complement given in Section 4 by concatenating the shifts of f_0 (corresponding to $[M_{00}\ M_{01}]$) to M in step 9 and rearranging the blocks such that the lower right one is occupied by $\begin{bmatrix}\hat{R}_b & \hat{Z}_1\\ 0 & \hat{R}_i\end{bmatrix}$. In fact, for any column permutation that gives a full rank $\begin{bmatrix}\hat{R}_b & \hat{Z}_1\\ 0 & \hat{R}_i\end{bmatrix}$, we can bring M into the form (5) and it is clear that

$$\mathbb{C}^{\#S(M)} \simeq R^h_t = \overline{I}_t \oplus \operatorname{span}(\mathcal{O}^h)$$

with $R^h = \mathbb{C}[x_0, x_1, \ldots, x_n]$ and $\mathcal{O}^h$ contains the homogenized monomials in $\mathcal{O}$ (they are homogenized to degree t). Also, $R^h_t = \overline{I}_t \oplus (R^h/\overline{I})_t$ so

$$\operatorname{span}(\mathcal{O}) \simeq \operatorname{span}(\mathcal{O}^h) \simeq \operatorname{span}(\mathcal{O}_M) \simeq (R/\overline{I})_t \simeq \mathbb{C}[x]/I.$$

These are isomorphisms of $\mathbb{C}$-vector spaces. The first one is given by homogenization and the last isomorphism follows from the genericity assumptions and the fact that the Hilbert polynomial stabilizes at regularity [10]. The isomorphism $\operatorname{span}(\mathcal{O}) \simeq \operatorname{span}(\mathcal{O}_M)$ is given by an invertible 'change of basis' matrix constructed as follows. The basis transformation $\mathcal{O}_M \to \mathcal{O}$ is a matrix with columns equal to the normal forms of $\mathcal{O}_M$ in $\mathcal{O}$. Call this matrix T. Then the multiplication maps m_{x_i} in $\mathcal{O}$ give multiplication maps m'_{x_i} in $\mathcal{O}_M$, given by $m'_{x_i} = T^{-1} m_{x_i} T$. This transformation makes $\operatorname{span}(\mathcal{O}) \simeq \operatorname{span}(\mathcal{O}_M) \simeq \mathbb{C}[x]/I$ isomorphisms of $\mathbb{C}$-algebras.

To show the connection with border bases, we will also work with $\mathcal{O}_M$ for simplicity. Any other border basis will do. Suppose we choose the basis $\mathcal{O}_M$ in steps 6, 7 to compute $m'_{x_1}, \ldots, m'_{x_n}$ in this basis. Note that this basis is connected to 1. Set $B = \operatorname{span}(\mathcal{O}_M)$ and consider the $\mathbb{C}$-linear map

$$N:\ B \cup \left(\bigcup_{i=1}^{n} x_i \cdot B\right) \longrightarrow B,$$
$$b \longmapsto \begin{cases} b & b \in B \\ m'_b(1) & b \notin B \end{cases}$$

where $m'_b = b(m'_{x_1}, \ldots, m'_{x_n})$. We show that $I = \langle \ker(N) \rangle$. Since m'_{x_i} represents multiplication with x_i modulo I, m'_b represents multiplication with b modulo I and we have that $\ker(N) \subset I$ and hence $\langle \ker(N) \rangle \subset I$. Let $K = (\bigcup_{i=1}^{n} x_i \cdot \mathcal{O}_M) \backslash \mathcal{O}_M$ and let g_α be the polynomial (4) for every $x^\alpha \in K$. Note that $\ker(N) = \operatorname{span}(g_\alpha, \alpha \in K) \subset I$. Any polynomial $f \in \mathbb{C}[x]$ can be written as $f = \sum_{\alpha \in K} c_\alpha g_\alpha + \overline{f}^{\mathcal{O}_M}$ with $c_\alpha \in \mathbb{C}[x]$ using a division algorithm as described in Chapter 4 of [5]. Since $\mathcal{O}_M$ is a (border) basis for $\mathbb{C}[x]/I$, $\overline{f}^{\mathcal{O}_M} = 0$ when $f \in I$. Therefore $I \subset \langle g_\alpha, \alpha \in K \rangle = \langle \ker(N) \rangle \subset I$ so $I = \langle g_\alpha, \alpha \in K \rangle = \langle \ker(N) \rangle$. It follows from Theorem 4 that $M_i(b) = N(x_i b) = m'_{x_i}(b)$ represents multiplication with x_i in the basis $\mathcal{O}_M$ for $\mathbb{C}[x]/I$ and the m'_{x_i} commute. For any other basis $\mathcal{O}$ with transformation matrix $T : \mathcal{O}_M \to \mathcal{O}$, commutativity of the resulting m_{x_i} follows from the relation $m_{x_i} = T m'_{x_i} T^{-1}$.

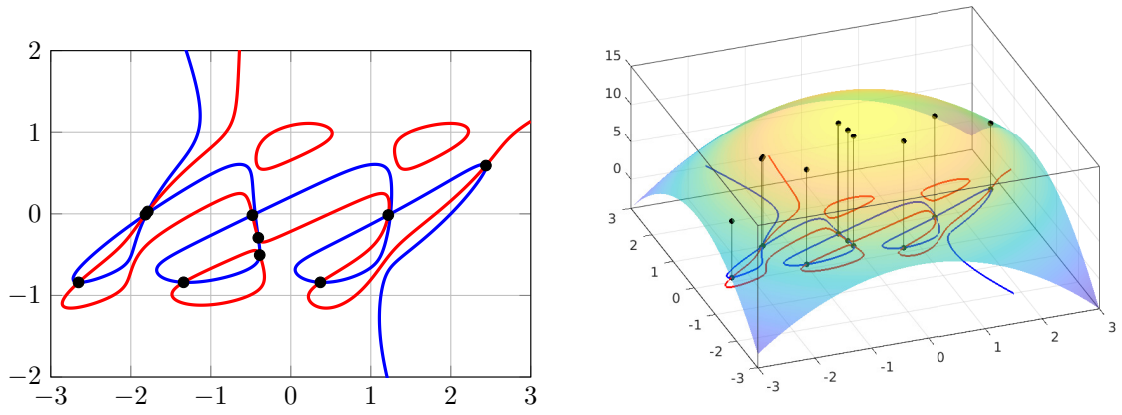

Figure 1: Left: zero level lines in $\mathbb{R}^2$ of two bivariate polynomials of degree 7 (—) and 6 (—) together with the real solutions (•). Right: The surface $f(x_1, x_2) = -(x_1^2 + x_2^2) + 0.1xy + 15$ and the real eigenvalues of m_f.

7 Numerical experiments

In this section, we use Algorithm 1 for some numerical experiments and compare it to Bertini [1, 2] and PHClab [15].

7.1 Evaluating a polynomial function on $\mathbb{V}(I)$

Theorem 3 implies that we can evaluate a function $f \in \mathbb{C}[x]$ on $\mathbb{V}(I)$ by calculating the eigenvalues of $m_f = f(m_{x_1}, \ldots, m_{x_n})$. Note that this expression for m_f is well defined because of the commutativity of the m_{x_i}. Algorithm 1 can be used if I satisfies the assumptions made in this paper. As a test of correctness, we have evaluated the quadric $f(x_1, x_2) = -(x_1^2 + x_2^2) + 0.1xy + 15$ on the variety defined by two bivariate polynomials of degree 7 and 6, shown in Figure 1. For the computed multiplication matrices, we compute

$$\frac{\|m_{x_1} m_{x_2} - m_{x_2} m_{x_1}\|_2}{\|m_{x_1} m_{x_2}\|_2} = 5.5552 \cdot 10^{-13}.$$

This shows that the multiplication matrices commute (up to 13 digits of accuracy).

7.2 Solving generic systems

We now use the obtained multiplication maps to compute the solutions $\mathbb{V}(I)$ of square systems of polynomial equations in the following way. We perform a simultaneous diagonalization of the identity matrix together with the n multiplication maps m_{x_i}. For this, we use the method `cpd_gevd` in Tensorlab [25, 17, 8]. We compare the results (accuracy and computation time) with the homotopy solvers BertiniLab [2] and PHClab [15]. To obtain the results, we used Matlab and we generated generic polynomials f in the following way. We fix a Newton polytope P of f and to every point in $P \cap \mathbb{Z}^n_{\geq 0}$ we assign a real number drawn from a normal distribution with $\mu = 0$ and $\sigma = 1$ (using the `randn` command in Matlab). These numbers are the coefficients of the monomials in $S(f)$. To measure the accuracy of the resulting multiplication matrices, we calculate the condition number of the matrix inverted in step 10 of Algorithm 1. The accuracy of a solution z of a square system $f_1 = \ldots = f_n = 0$ is measured by the *residual*.

Definition 2. *Given a square system of polynomial equations $f_1 = \ldots = f_n = 0$ with $f_1, \ldots, f_n \in \mathbb{C}[x]$ and a point $z \in \mathbb{C}^n$. The* residual *r of z is defined as*

$$r_i = \frac{|f_i(z)|}{f_{i,\mathrm{abs}}(z_{\mathrm{abs}}) + 1}, \qquad r = \frac{1}{n}\sum_{i=1}^{n} r_i,$$

where $|\cdot|$ denotes the absolute value, $f_{i,\mathrm{abs}}$ is f_i where the coefficients $c_{\alpha,i}$ of f_i are replaced by their absolute values and z_{abs} is the point in $\mathbb{C}^n$ obtained by taking the absolute values of all the components of z.

The term $+1$ in the denominator of the r_i makes it clear that we are using a mixed relative and absolute criterion, to take into account the possibility that $f_{i,\mathrm{abs}}(z_{\mathrm{abs}}) = 0$.

We first investigate the influence of the automated choice of basis made in our algorithm. We compare it to the fixed choice of the block basis given in (3). This is the basis that is used (implicitly) in root finding using u-resultants [7, Chapter 3]. We first check that it is not just the block basis itself that comes out of our algorithm. We generated two random dense polynomials $f_1, f_2 \in \mathbb{C}[x_1, x_2]$ of degree $d_1 = d_2 = 10$. The support of the associated dense Macaulay matrix M is all monomials of degree up to $d_1 + d_2 - 1 = 19$. The basis $\mathcal{O}$ should count 100 elements (Theorem 1). Figure 2 shows that, indeed, the choice of basis is significantly different. Note also that the resulting basis does not have the connected to 1 property.

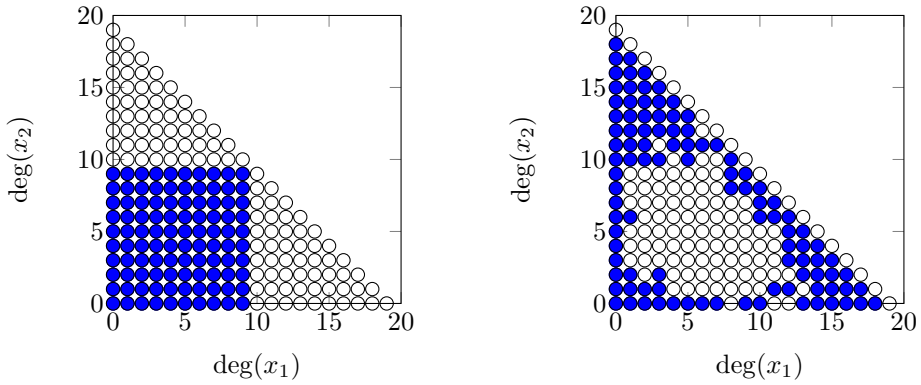

Figure 2: Left: the block basis $\mathcal{O}$ given in (3). Right: the basis $\mathcal{O}$ chosen by Algorithm 1. Black circles indicate the support $S(M)$ of the Macaulay matrix.

We now check the accuracy of the multiplication matrices by computing the condition number of the coefficient matrix inverted in step 10. For a condition number of order 10^l, we expect to loose l accurate digits w.r.t. the machine precision. Figure 3 shows the results for bivariate systems of increasing degree[3] up to 20. By using the QR decomposition with optimal column pivoting the condition number is controlled and it gets no larger than $\pm 10^4$. With our machine precision of order 10^{-16} (double precision), this means that the forward error on the multiplication matrices is of order 10^{-12}. For the same set of generic bivariate systems of degree 1 up to 20 we also calculated the maximal residual of all of the calculated solutions. This is shown in the right part of Figure 3. One can expect that more accurate multiplication maps lead to more accurate solutions, which is confirmed by the figure. For degrees higher than 15, the solutions obtained using the block basis no

[3]By degree d we mean here that both polynomials f_1 and f_2 are generic of degree d.

longer made sense. The results are avaraged out over 20 experiments. These results clearly show that a numerically justified choice of basis is crucial for the feasibility of normal form algorithms to compute multiplication matrices.

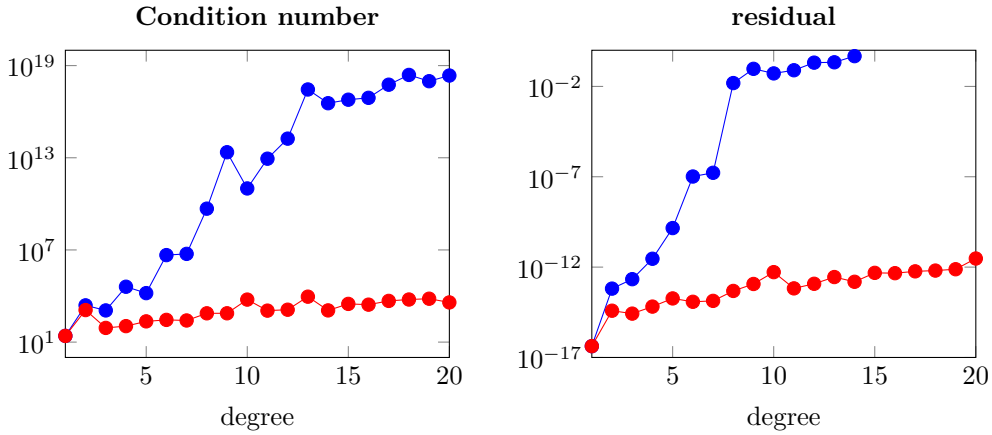

Figure 3: Left: condition number for the computation of the multiplication matrices with block basis (—●—) and smart choice of basis (—●—) for bivariate systems of increasing degree. Right: Maximal residual with the block basis (—●—) and the QR choice of basis (—●—) for the same systems.

In the following, we only work with the automated choice of basis. Some results for dense systems with more variables are shown in Figure 4. The figure shows that even for large systems, all solutions are found with a small residual. For example, in the case $n = 3$ with degree 21, there are 9261 solutions in $\mathbb{C}^3$, all found with a residual smaller than 10^{-10}. We also note that the residual would drop to machine precision after one 'refining' iteration of Newton's method.

As for the computation time, the figure shows that the method is very sensitive to the number of variables (it suffers from the 'curse of dimensionality'). The asymptotic complexity is $O(d^{3n})$, where d is the degree. Intuitively, we find the coordinates of the d^n solutions as eigenvalues and the cost of the algorithm is the number of eigenvalues cubed.

We compare our method to the Matlab interfaces of the homotopy continuation packages Bertini [2] and PHCpack [15][4]. The results are shown in Figure 5. The figure confirms that the complexity of our method grows drastically with n. For $n = 2$, however, we are slightly faster for degrees at least up to 25. In all figures, the residuals of our computed solutions are slightly bigger than the ones from the homotopy methods. This is because these methods intrinsically make use of Newton-Raphson refinement. One Newton sweep over our solutions would lead to a residual of order machine precision as well, because of the quadratic convergence property. An important remark is that continuation methods do not return *all* solutions in all cases. The methods might give up on certain paths along the way if the algorithm decides that the path seems to be diverging to infinity or if it crosses an ill-conditioned region. For $n = 2$ and degree 20, PHClab returns 398 solutions (2 solutions are lost) within slightly less than 4 seconds. For $n = 2$, degree 40, it takes 57 seconds to find 1575 out of the 1600 solutions. Using Bertini with double precision arithmetic [1], we find all solutions for $n = 2$, degree 20 within 12 seconds and 1587 out of 1600 solutions for $n = 2$, degree 40 within 350 seconds.

[4]We used default settings for both solvers.

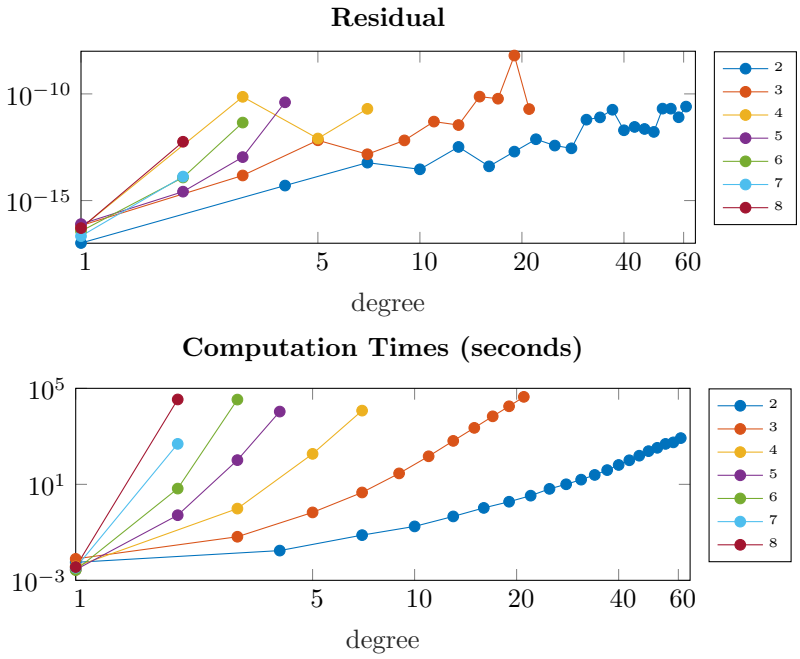

Figure 4: Maximal residual and computation times for systems of increasing degree with $n = 2, \ldots, 8$.

8 Conclusion and future work

We have presented a first normal form algorithm for zero-dimensional ideals coming from square polynomial systems that makes an automated, numerically justified choice of monomial basis for $\mathbb{C}[x]/I$ under certain genericity assumptions on I. Our numerical experiments show that this choice of basis makes it possible to perform the normal form computation in finite precision, while it can go terribly wrong by manually choosing a basis. Some ideas for future work are:

- Relaxing the genericity assumptions. What if the polynomials $f_1, \ldots, f_n$ are sparse?
- Solutions at infinity lead to linear dependencies in the columns of M_b, but it also causes the dimension of $\mathbb{C}[x]/I$ to drop. This can be incorporated in the algorithm.
- For multiple solutions of a square polynomial system, the canonical polyadic decomposition does not work. The coupling between the different coordinates can be made by using the left eigenvectors of the multiplication maps.
- The implementation is done in Matlab and a lot of computation time is spent on the construction of the Macaulay matrix M. We believe that an implementation in Julia, C(++), Fortran, ... could be a significant improvement.
- Taking all possible monomial multiples of the input equations to construct M leads to a number of redundant rows. This number becomes large very quickly when the number of variables increases. To reduce the computational cost and to enhance the performance for a larger number of variables, only a selection of the monomial multiples can be used. In doing so, a trade-off between speed and accuracy should be taken into account.

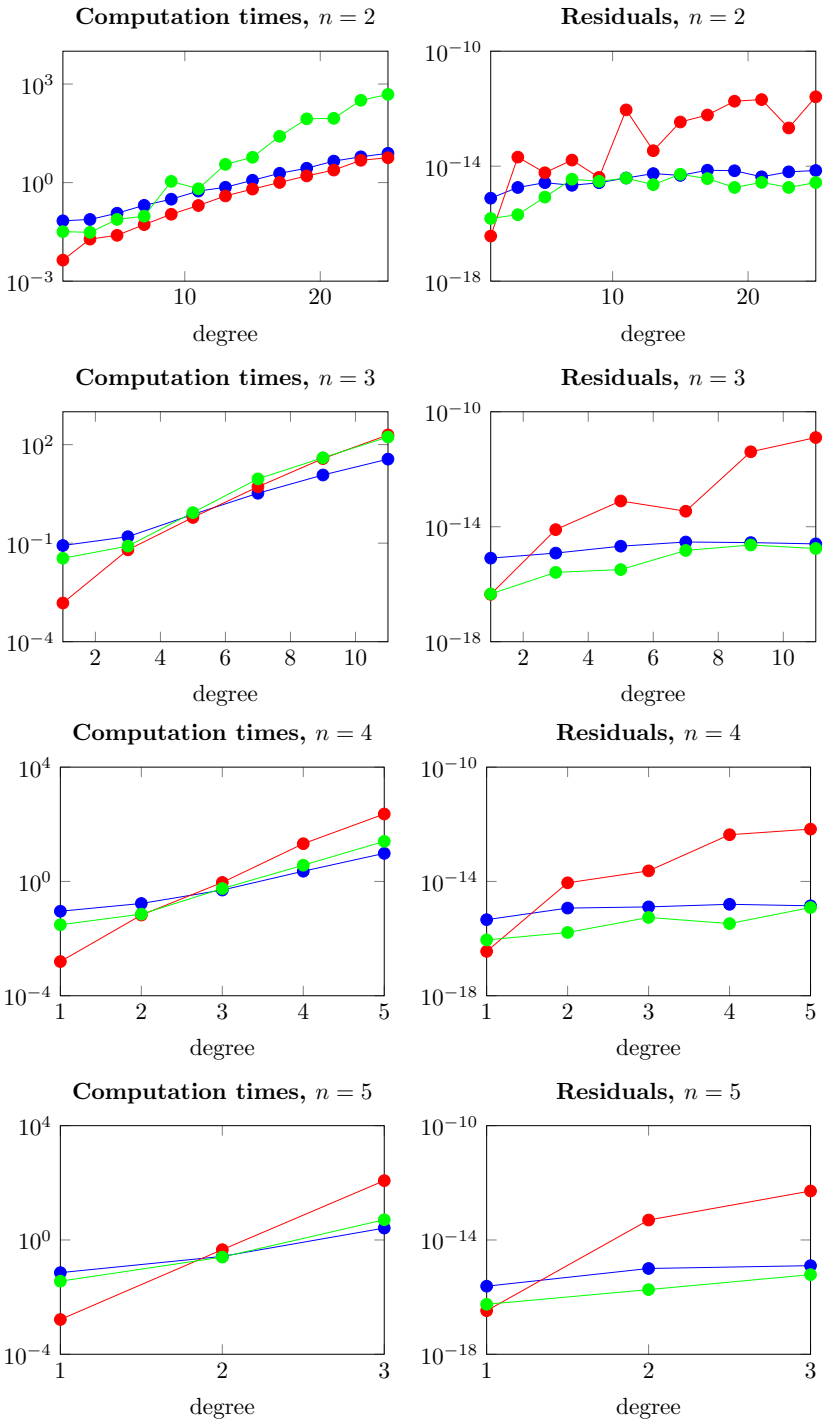

Figure 5: Comparison of the results for PHClab (—●—), BertiniLab (—●—) and our method (—●—) for $n = 2, 3, 4, 5$.

Acknowledgements

We want to thank David Cox for his useful comments and Bernard Mourrain for fruitful conversations.

References

[1] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. *Numerically solving polynomial systems with Bertini*, volume 25. SIAM, 2013.

[2] D. J. Bates, A. J. Newell, and M. Niemerg. Bertinilab: A MATLAB interface for solving systems of polynomial equations. *Numerical Algorithms*, 71(1):229–244, 2016.

[3] K. Batselier. *A Numerical Linear Algebra Framework for Solving Problems with Multivariate Polynomials*. KU Leuven - Faculty of Engineering Science, 2013. PhD thesis, promotor: Bart De Moor.

[4] B. Buchberger and F. Winkler. *Gröbner bases and applications*, volume 251. Cambridge University Press, 1998.

[5] E. Cattani, D. A. Cox, G. Chèze, A. Dickenstein, M. Elkadi, I. Z. Emiris, A. Galligo, A. Kehrein, M. Kreuzer, and B. Mourrain. Solving polynomial equations: foundations, algorithms, and applications (Algorithms and Computation in Mathematics). 2005.

[6] D. A. Cox, J. Little, and D. O'shea. *Ideals, varieties, and algorithms*, volume 3. Springer, 1992.

[7] D. A. Cox, J. Little, and D. O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[8] L. De Lathauwer. A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization. 28(3):642–666, 2006.

[9] P. Dreesen. *Back to the Roots*. KU Leuven - Faculty of Engineering Science, 2013. PhD thesis, promotor: Bart De Moor.

[10] D. Eisenbud. *The Geometry of Syzygies: A Second Course in Commutative Algebra and Algebraic Geometry*. Springer, New York, NY, 2005. OCLC: 249751633.

[11] M. Elkadi and B. Mourrain. *Introduction à la résolution des systèmes polynomiaux*, volume 59 of *Mathématiques et Applications*. Springer, 2007.

[12] I. Z. Emiris and J. Canny. A practical method for the sparse resultant. *Proc. ACM Intern. Symp. on Symbolic and Algebraic Computation*, pages 183–192, 1993.

[13] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of pure and applied algebra*, 139(1):61–88, 1999.

[14] G. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3):206–216, 1965.

[15] Y. Guan and J. Verschelde. PHClab: a MATLAB/Octave interface to PHCpack. *IMA Volumes in Mathematics and its Applications*, 148:15, 2008.

[16] R. Hartshorne. *Algebraic Geometry*. Springer, 1977.

[17] S. E. Leurgans, R. T. Ross, and R. B. Abel. A decomposition for three-way arrays. 14(4):1064–1083, 1993.

[18] F. S. Macaulay. *The algebraic theory of modular systems*. Cambridge University Press, 1994.

[19] B. Mourrain. A New Criterion for Normal Form Algorithms. In *Proceedings of the 13th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, LNCS, pages 430–443, London, UK, 1999. Springer-Verlag.

[20] B. Mourrain. Pythagore's dilemma, symbolic-numeric computation, and the border basis method. In *Symbolic-Numeric Computation*, pages 223–243. Springer, 2007.

[21] B. Mourrain and P. Trebuchet. Generalized normal forms and polynomial system solving. In *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*, pages 253–260. ACM, 2005.

[22] B. Mourrain and P. Trébuchet. Stable normal forms for polynomial system solving. *Theoretical Computer Science*, 409(2):229–240, 2008.

[23] H. J. Stetter. *Numerical Polynomial Algebra*. Society for Industrial and Applied Mathematics, 2004.

[24] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software (TOMS)*, 25(2):251–276, 1999.

[25] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab 3.0. *available online, URL:* `www.tensorlab.net`, 2016.