Solving Polynomial Systems Efficiently and Accurately

Bernard Mourrain, Simon Telen, Marc van Barel

SOLVING POLYNOMIAL SYSTEMS EFFICIENTLY AND ACCURATELY

BERNARD MOURRAIN, SIMON TELEN, AND MARC VAN BAREL*

Abstract. We consider the problem of finding the isolated common roots of a set of polynomial functions defining a zero-dimensional ideal I in a ring R of polynomials over $\mathbb{C}$. Normal form algorithms provide an algebraic approach to solve this problem. We use new characterizations of normal forms and describe accurate and efficient constructions that allow us to compute the algebra structure of R/I, and hence the solutions of I. We show how the resulting algorithms give accurate results in double precision arithmetic and compare with normal form algorithms using Groebner bases and homotopy solvers.

1. Introduction

Several problems in science and engineering boil down to the problem of finding the (isolated) common roots of a set of multivariate polynomial equations. In mathematical terms, if $R = \mathbb{C}[x_1, \ldots, x_n]$ is the ring of polynomials over $\mathbb{C}$ in the n indeterminates $x_1, \ldots, x_n$ and $I = \langle f_1, \ldots, f_s \rangle \subset R, f_i \in R$, the problem can be formulated as finding the points in the algebraic set $\mathbb{V}(I) = \{z \in \mathbb{C}^n : f_i(z) = 0, i = 1, \ldots, s\} = \{z \in \mathbb{C}^n : f(z) = 0, \forall f \in I\}$.
There exist several methods to find all the roots of a set of polynomial equations [30, 4]. The most important classes are homotopy continuation methods [1, 34], subdivision methods [24] and algebraic methods [12, 27, 7, 9, 22, 33]. When the roots of the ideal I are isolated points (I is called *zero-dimensional* in this case), eigenvalue, eigenvector techniques can be used to find these roots [7, 11, 13, 28]. One can find the origins of these techniques in ancient works on resultants by Bézout, Sylvester, Cayley, Macaulay.... Explicit constructions of matrices of polynomial coefficients are exploited to compute projective resultants of polynomial systems (see e.g., [19, 20]). These matrix constructions have been further investigated to compute other types of resultants such as toric or sparse resultants [7, 12, 8] or residual resultants [3]. See e.g. [13] for an overview of these techniques. The key observation to translate the root finding problem into a linear algebra problem is a standard result in algebraic geometry that states that R/I is finitely generated over $\mathbb{C}$ as a $\mathbb{C}$-algebra (it is a finite dimensional $\mathbb{C}$-vector space) if and only if I is zero-dimensional. Moreover, $\dim_{\mathbb{C}}(R/I) = \delta$, where δ is the number of points defined by I, counting multiplicities. See for instance [6, Theorem 6, Chapter 5, §3]. The map $M_f : R/I \to R/I : g + I \mapsto fg + I$, representing 'multiplication by $f + I$' in R/I is linear. Fixing a basis for R/I, M_f is a $\delta \times \delta$ matrix. A well known result is that the eigenvalue structure of such *multiplication matrices* reveals the coordinates of the points in $\mathbb{V}(I)$ [11, 7, 28].
In general, normal form algorithms execute the following two main steps.

(1) Compute the multiplication matrices $M_{x_1}, \ldots, M_{x_n}$ with respect to a suitable basis of R/I.
(2) Compute the points $\mathbb{V}(I)$ from the eigenvalue structure of these matrices.

*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 EOS Project 30468160 (Structured Low-Rank Matrix/Tensor Approximation: Numerical Optimization-Based Algorithms and Applications), and by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization).

In this paper we focus on step (1). Once a basis $\mathcal{B} = \{b_1 + I, \ldots, b_\delta + I\}$ of R/I is fixed, the i-th column of M_{x_j} corresponds to the coordinates of $x_j b_i + I$ in $\mathcal{B}$. These coordinates are found by projecting $x_j b_i$ onto $B = \mathrm{span}(b_1, \ldots, b_\delta)$ along I. A well-known method to compute this projection map uses Groebner bases with respect to a certain monomial ordering [6, 7]. The resulting basis consists of monomials and it follows from the monomial ordering that is chosen. The monomial basis is sensitive to perturbations of the input coefficients. Also, Groebner basis computations are known to be unstable and hence unfeasible for finite precision arithmetic. Border bases have been developed to enhance robustness and efficiency of the Groebner basis normal form method [22, 25, 23, 26]. There is more freedom in choosing the basis. However, because of the incremental nature of border basis algorithms, it is not clear how to make a good choice of basis that takes the global numerical properties of the input equations into account. In [33] it is shown that the choice of basis for R/I can be crucial for the accuracy of the computed multiplication maps and a monomial basis $\mathcal{B}$ is chosen using column pivoted QR factorization on a large Macaulay-type matrix for solving generic, dense problems. In [32] a general algebraic framework is proposed for constructing normal forms with respect to a numerically justified basis for R/I using similar numerical linear algebra techniques. Some algorithms are proposed for finding affine, toric, homogeneous and multihomogeneous roots from multiplication tables. The resulting bases consist of monomials, although the theorems allow much more general constructions.
Here, we show how the results from [32] lead to other normal forms, arising from the kernel of a matrix with rows corresponding to polynomials in a vector subspace $V = I \cap V$ of I. In the next section, we introduce truncated normal forms and we summarize their relation to the main results in [32]. In Section 3 we discuss truncated normal form constructions. In Section 4 we describe new constructions of truncated normal forms using evaluations of the polynomials and different bases for the vector spaces involved in the algorithm. We consider in particular orthogonal polynomial bases such as the Chebyshev basis, since they have interesting interpolation and approximation properties. Section 5 shows some numerical experiments and contains a comparison of our normal form algorithms with homotopy solvers and Groebner basis methods.

2. Truncated Normal Forms

Throughout the paper, let $R = \mathbb{C}[x_1, \ldots, x_n]$ be the ring of polynomials over $\mathbb{C}$ in the variables $x_1, \ldots, x_n$ and let $I \subset R$ be a zero-dimensional ideal such that $\dim_{\mathbb{C}}(R/I) = \delta$, the number of points defined by I, counting multiplicities.

Definition 2.1. *A* normal form *on R w.r.t. I is a linear map $\mathcal{N} : R \to B$ where $B \subset R$ is a vector subspace of dimension δ over $\mathbb{C}$ such that*

$$0 \longrightarrow I \longrightarrow R \xrightarrow{\mathcal{N}} B \longrightarrow 0$$

is exact and $\mathcal{N}_{|B} = \mathrm{id}_B$.

The aim in this paper is to construct normal forms restricted to finite dimensional subspaces of R with respect to I given a set of generators of I and to exploit them for solving zero-dimensional polynomial systems. The following is the main result (Theorem 3.1) of [32].

Theorem 2.2. *Let V be a finite dimensional vector subspace of R and $N : V \to \mathbb{C}^\delta$ a linear map with $\ker N \subset I \cap V$. If there is $u \in V$ such that $u + I$ is a unit in R/I and V contains a vector subspace $W \subset V$ such that $x_i \cdot W \subset V, i = 1, \ldots, n$ and $N_{|W} : W \to \mathbb{C}^\delta$ is surjective, then for any subspace $B \subset W$ of dimension δ for which $W = B \oplus \ker N_{|W}$ we have:*

(i) $N^* = N_{|B}$ is invertible,
(ii) there is an isomorphism of R-modules $\phi : B \to R/I$,
(iii) $V = B \oplus V \cap I$ and $I = (\langle \ker(N) \rangle : u)$,

(iv) the maps N_i given by

$$\begin{aligned} N_i : B &\longrightarrow \mathbb{C}^\delta, \\ b &\longrightarrow N(x_i b) \end{aligned}$$

for $i = 1, \ldots, n$ can be decomposed as $N_i = N^* \circ \tilde{M}_{x_i}$ where $\tilde{M}_{x_i} = \phi^{-1} \circ M_{x_i} \circ \phi$ define the multiplications by x_i in B modulo I via the isomorphism of point (ii).

Proof. For a detailed proof we refer to [32]. □

It follows from Theorem 2.2 that $\mathcal{N}_{|V} : V \to B : v \mapsto (N^*)^{-1} \circ N(v)$ is the restriction of a normal form on R w.r.t. I onto B. We will call N a *Truncated Normal Form* (TNF). The map N^* gives the isomorphism $B \simeq \mathbb{C}^\delta$ by defining coordinates on B. The isomorphism ϕ is given by $\phi(f) = fu + I$, so that if $1 \in V$ we can take $u = 1$ and we have that ϕ just sends a polynomial to its residue class in R/I. Given a basis $\mathcal{B} = \{b_1, \ldots, b_\delta\}$ of B, the matrices of $(N^*)^{-1} N_i = \tilde{M}_{x_i}$ are the matrices of the multiplication maps M_{x_i} with respect to the basis $\{\phi(b_1), \ldots, \phi(b_\delta)\}$ of R/I. Therefore, the problem of computing multiplication maps (referred to as step (1) of any normal form algorithm earlier) is reduced to finding N, V and $W \subset V$ with the right properties.

2.1. **TNF for homogeneous ideals.** We consider a system of s homogeneous polynomials $f_1, \ldots, f_s \in \mathbb{C}[x_0, x_1, \ldots, x_n]$ such that $\mathbb{V}(f_1, \ldots, f_s)$ is a finite set of projective roots in $\mathbb{P}^n$. We are interested in finding all these projective roots. Denote $S = \mathbb{C}[x_0, x_1, \ldots, x_n]$ and let $I = \langle f_1, \ldots, f_s \rangle \subset S$ be a zero-dimensional homogeneous ideal generated by these homogeneous polynomials in $n+1$ variables with $\delta < \infty$ solutions in $\mathbb{P}^n$, counting multiplicities. For $h \in S_1$ such that h is a non zero divisor in S/I, define $M_{x_i} : S_d/I_d \to S_d/I_d : f + I_d \to (x_i/h)f + I_d$. The eigenvalues of M_{x_i} are the evaluations of x_i/h in the points of $\mathbb{V}(I) \subset \mathbb{P}^n$ (it is easy to check that this is well-defined).

Theorem 2.3. *Let $V = S_d$, a finite dimensional vector subspace of S and $N : V \to \mathbb{C}^\delta$ a linear map with $\ker N \subset I \cap V$. Take $h \in S_1$ such that $h + I$ is a unit in S/I and define $W = S_{d-1}$. If $N_h : W \to \mathbb{C}^\delta : w \mapsto N(hw)$ is surjective, then for any subspace $B \subset W$ of dimension δ for which $W = B \oplus \ker N_h$ we have:*

(i) $N^* = (N_h)_{|B}$ is invertible,
(ii) there is an isomorphism of $\mathbb{C}[\frac{x_0}{h}, \ldots, \frac{x_n}{h}]$-modules $\phi : h \cdot B \to S_d/I_d$,
(iii) $S_k = h^{k-d+1} \cdot B \oplus I_k$ and $I = (\langle \ker(N) \rangle : h^\infty)$,
(iv) the maps N_i given by

$$\begin{aligned} N_i : B &\longrightarrow \mathbb{C}^\delta, \\ b &\longrightarrow N(x_i b) \end{aligned}$$

for $i = 1, \ldots, n$ can be decomposed as $N_i = N^* \circ \tilde{M}_{x_i}$ where $\tilde{M}_{x_i} = \phi^{-1} \circ M_{x_i} \circ \phi$ define the multiplications by x_i/h in $h \cdot B$ modulo I_d via the isomorphism of point (ii).

Proof. For a detailed proof we refer to [32]. □

To show how Theorem 2.3 follows from Theorem 2.2, we use the fact that after applying a generic change of coordinates on $\mathbb{P}^n$, all projective solutions lie in an affine chart and they can be found as solutions in $\mathbb{C}^n$. Let $R = \mathbb{C}[y_1, \ldots, y_n]$ be the ring of polynomials in n variables. Consider the homogenization isomorphisms

$$\begin{aligned} \sigma_d : R_{\leq d} &\longrightarrow S_d, \\ f &\longrightarrow h^d f\left(\frac{x_1}{h}, \ldots, \frac{x_n}{h}\right) \end{aligned}$$

for every $d \in \mathbb{N}$. The inverse dehomogenization map in degree d is given by

$$\begin{aligned}\sigma_d^{-1} : S_d &\longrightarrow R_{\leq d},\\ f &\longrightarrow f\left(\frac{1-\sum_{i=1}^n h_i y_i}{h_0}, y_1, \ldots, y_n\right).\end{aligned}$$

Its definition is independent of the degree d, so that we can omit d and denote it σ^{-1}. The ideal $\tilde{I} = \langle \sigma^{-1}(f_1), \ldots, \sigma^{-1}(f_n)\rangle$ has δ solutions in $\mathbb{C}^n$, counting multiplicities (since h is a non zero divisor in S/I). Let $\tilde{V} = R_{\leq d}$ and $\tilde{W} = R_{\leq d-1}$. The map $\tilde{N} : \tilde{V} \to \mathbb{C}^\delta$ given by $\tilde{N} = N \circ \sigma_d$ is surjective and $\ker(\tilde{N}) \subset \tilde{I} \cap \tilde{V}$. Also, $y_i \cdot \tilde{W} \subset \tilde{V}$, $i = 1, \ldots, n$. For $f \in R_{\leq d-1}$, $\sigma_d(f) = h\sigma_{d-1}(f)$. Therefore $\tilde{N}(R_{\leq d-1}) = N(h \cdot S_{d-1})$ and $\tilde{N}_{|\tilde{W}} = N_h \circ \sigma_{d-1}$ is surjective. Theorem 2.2 now applies and gives a subspace $R/\tilde{I} \simeq \tilde{B} \subset \tilde{W}$ with isomorphism $\tilde{\phi} : \tilde{B} \to R/\tilde{I}$. The isomorphism ϕ from point (ii) of Theorem 2.3 is then given by $\phi = \sigma_d \circ \tilde{\phi}$. If M_{y_i} are the multiplication operators in $R/\tilde{I}$, then

$$\begin{aligned} M_{x_i} &= \sigma_d \circ M_{y_i} \circ \sigma^{-1},\\ M_{x_0} &= \sigma_d \circ g(M_{y_1}, \ldots, M_{y_n}) \circ \sigma^{-1}\end{aligned}$$

with $g(y_1, \ldots, y_n) = \frac{1-\sum_{i=1}^n h_i y_i}{h_0}$.
Since the matrices $(N^*)^{-1}N_i$ commute, for an eigenvalue $\lambda_i = \frac{z_{ji}}{h(z_j)}$ of M_{x_i} and $\lambda_k = \frac{z_{jk}}{h(z_j)}$ of M_{x_k} with common eigenvector v, we have:

$$\lambda_k (N^*)^{-1} N_i v = \lambda_k \lambda_i v = \lambda_i (N^*)^{-1} N_k v.$$

Left multiplication by N^* gives $\lambda_k N_i v = \lambda_i N_k v$ and the generalized eigenvalues of $N_i v = \lambda N_k v$ are the fractions z_{ji}/z_{jk}. This means that we do not need to construct N^* to find the projective coordinates of the solutions, as long as we have $N_i, i = 0, \ldots, n$ and a generic linear combination of the N_i is invertible.

2.2. **TNF for toric ideals.** Theorem 2.2 can be applied to find the toric solutions of a zero-dimensional ideal $I = \langle f_1, \ldots, f_s\rangle$ in the ring of multivariate Laurent polynomials $R_{x_1 \cdots x_n} = \mathbb{C}[x_1^{\pm 1}, \ldots, x_n^{\pm 1}] \supset R$. We can assume that $f_i \in R$, since any monomial is invertible in $R_{x_1 \cdots x_n}$. The roots of I are the points in $\mathrm{Specm}(R_{x_1 \cdots x_n}) = (\mathbb{C}\backslash\{0\})^n$ in which all of the f_i vanish. Define $I^* = I \cap R$, which is an ideal in R with roots equal to the roots of I in the algebraic torus $(\mathbb{C}\backslash\{0\})^n$. We can apply Theorem 2.2 to compute multiplication tables in R/I^*, which suffices to find all these points.

3. Computing Truncated Normal Forms

In this section, we describe how a Truncated Normal Form N can be constructed from a given set of generators. We define the spaces $W \subset V \subset R$ satisfying the assumptions of the theorems in different cases. We do not fix any bases for the involved vector spaces yet. After doing so in the next section, we can work with matrices and show some results in Section 5. Unlike the approach in [32], we first specify a general 'template' algorithm and specify the parameters and maps in the subsections for the dense affine, sparse affine and projective cases respectively.
We consider an ideal $I \subset R$ generated by $f_1, \ldots, f_s \in R$. We are going to construct the truncated normal form N using maps of the form:

$$\begin{aligned}\mathcal{R}_{f_1,\ldots,f_s} : \quad V_1 \times \cdots \times V_s &\longrightarrow V\\ (q_1, \ldots, q_s) &\longmapsto q_1 f_1 + \cdots + q_s f_s.\end{aligned} \tag{1}$$

where $V_1, \ldots, V_s, V$ are vector subspaces of R such that $f_i \cdot V_i \subset V$. Hereafter, we will consider vector spaces span by monomials. Let W be a vector subspace of R such that $x_i \cdot W \subset V$.

Algorithm 1 computes the matrix N, from the null space of $\mathcal{R}^T$, a basis $\mathcal{B}$ in the quotient algebra R/I defining the solutions of $f_1 = 0, \ldots, f_s = 0$, and the matrices M_{x_i} of multiplication by the variables x_i in the basis $\mathcal{B}$.

Algorithm 1 Computes the structure of the algebra R/I

1: **procedure** AlgebraStructure($f_1, \ldots, f_s$)
2: $\mathcal{R} \leftarrow \mathcal{R}_{f_1,\ldots,f_s}$
3: $N \leftarrow \text{null}(\mathcal{R}^\top)^\top$
4: Choose $h = h_0 + h_1 x_1 + \ldots + h_n x_n$ such that $h \cdot W \subset V$
5: Let $N_0 : w \in W \mapsto N(hw)$
6: $N^* \leftarrow$ an invertible submatrix of N_0
7: $\mathcal{B} \leftarrow$ monomials corresponding to the columns of N^*
8: **for** $i = 1, \ldots, n$ **do**
9: $N_i \leftarrow$ columns of N corresponding to $x_i \cdot \mathcal{B}$
10: $M_{x_i} \leftarrow (N^*)^{-1} N_i$
11: **end for**
12: **return** $M_{x_1}, \ldots, M_{x_n}$
13: **end procedure**

By construction, we have that $\text{null}(N) = \text{im}(\mathcal{R}_{f_1,\ldots,f_s}) \subset I \cap V$. To apply Theorem 2.2, we need to check that $\text{rank}(N_{|W}) = \text{rank}(N_0) = \delta$, where δ is the number of points defined by I, counting multiplicities.

In step 6 of Algorithm 1, any invertible submatrix of N_0 can be selected. Since we are working in finite precision, it is important for the accuracy of the computations that the resulting N^* be a submatrix with an inverse that can be computed accurately. Therefore, we heuristically minimize its condition number over all submatrices by performing a column pivoted QR factorization on N_0, similar to what is done in [33]. The result is a monomial basis for R/I with good numerical properties.

We are now going to analyze cases for which this algorithm provides the structure of the quotient algebra.

3.1. **Generic dense square systems.** We consider first the case of a square and generic system $\{f_1, \ldots, f_n\}$, in the sense that there are $\delta = \prod_{i=1}^n \deg(f_i)$ solutions, counting multiplicities, in $\mathbb{C}^n$. We denote these solutions by $\mathbb{V}(I) = \{z_1, \ldots, z_{\delta_0}\} \subset \mathbb{C}^n$, where $\delta_0 \leq \delta$ is the number of distinct solutions.

Let $d_i = \deg(f_i)$, $\rho = \sum_{i=1}^n d_i - n + 1$, let $V = R_{\leq \rho}$ be the space of polynomials of degree $\leq \rho$ and $V_i = R_{\leq \rho - d_i}$. We take $W = R_{\leq \rho - 1}$ so that $x_i \cdot W \subset V$, and $h = 1$.

We check that $\text{rank}(N_{|W}) = \text{rank}(N_0) = \delta$. We use the classical Macaulay resultant matrix construction defined as follows, for a generic polynomial f_0 of degree 1.

$$\begin{array}{rccc} \mathcal{R}_0 : & V_0 \times V_1 \times \cdots \times V_n & \longrightarrow & V \\ & (q_0, q_1, \ldots, q_n) & \longmapsto & q_0 f_0 + q_1 f_1 + \cdots + q_n f_n. \end{array}$$

A square submatrix $\mathcal{R}'$ of the matrix of $\mathcal{R}_0$ such that $\det(\mathcal{R}') \neq 0$ is a nontrivial multiple of the resultant $\text{Res}(f_0, f_1, \ldots, f_n)$ [7, 20]. In the notation of [33], the monomial multiples of f_0 involved in $\mathcal{R}'$ are with exponents in $\Sigma_0 = \{\alpha \in \mathbb{N}^n : \alpha_i < d_i, i = 1, \ldots, n\}$. The set $\mathcal{B}_0$ of monomials with exponents in Σ_0 corresponds generically to a basis (the so-called Macaulay basis) of R/I: $B_0 = \text{span}(\mathcal{B}_0) \simeq R/I$. The matrix $\mathcal{R}'$ decomposes as

$$\mathcal{R}' = \begin{bmatrix} \mathcal{R}_{00} & \mathcal{R}_{01} \\ \mathcal{R}_{10} & \mathcal{R}_{11} \end{bmatrix}$$

where the rows and columns of the first block $\mathcal{R}_{00}$ are indexed by $\mathcal{B}_0$. The matrix $\tilde{\mathcal{R}} = \begin{bmatrix}\mathcal{R}_{01}\\ \mathcal{R}_{11}\end{bmatrix}$ representing monomial multiples of $f_1, \ldots, f_n$ is such that $\mathrm{im}(\tilde{\mathcal{R}}) \subset I \cap V$. Since for generic systems $f_1, \ldots, f_n$, the matrix $\mathcal{R}_{11}$ is invertible (see [20], [7, Chapter 3]), the rank of $\tilde{\mathcal{R}}$ is $\dim V - \delta$. This implies that the null space of $\tilde{\mathcal{R}}^\top$ is of dimension δ, i.e. $\mathrm{rank}(N) = \delta$. We deduce from Theorem 2.2, that N is the restriction of a normal form along $I = \langle f_1, \ldots, f_n\rangle$ and $\tilde{M}_{x_i}$ are the multiplication tables by the variables x_i in the basis $\mathcal{B}$ of R/I.

3.2. **Projective zero-dimensional systems.** The following result from [32] shows that the hypotheses of Theorem 2.3 can be fulfilled for d greater than or equal to the regularity and provides a new criterion for detecting the d-regularity of a projective zero-dimensional ideal. We recall that the regularity $\mathrm{reg}(I)$ of an ideal I is $\min(d_{i,j} - i)$ where $d_{i,j}$ are the degrees of generators of the i^{th}-syzygy module in a minimal resolution of I (see [10]). An ideal is d-regular if $d \geq \mathrm{reg}(I)$.

Proposition 3.1. *Let I be a homogeneous ideal with $\delta < \infty$ solutions in $\mathbb{P}^n$, counting multiplicities. The following statements are equivalent:*

(i) There is a linear map $N : S_d \to \mathbb{C}^\delta$ with $\ker(N) \subset I \cap S_d$ and $N_h : S_{d-1} \to \mathbb{C}^\delta$ given by $N_h(f) = N(h \cdot f)$ is surjective for generic h,

(ii) I is d-regular.

Proof. See [32]. □

It follows that if $I = (f_1, \ldots, f_s) \subset S$ is homogeneous zero-dimensional and if we choose $d \geq \mathrm{reg}(I)$ and construct $N = \ker \mathcal{R}^\top_{f_1,\ldots,f_s}$ where $\mathcal{R}_{f_1,\ldots,f_s}$ is the map defined in (1) with $V_i = S_{d-\deg(f_i)}$, $V = S_d, W = S_{d-1}$, then the hypothesis of Theorem 2.3 is satisfied. As was known by Macaulay [20], for generic square homogeneous systems $(f_1, \ldots, f_n)$, we have $\rho := \mathrm{reg}(I) = \sum_{i=1}^n d_i - n + 1$ where $d_i = \deg(f_i)$. In this case, the construction of N is the homogenized version of the construction in Subsection 3.1.

3.3. **Generic sparse square systems.** The sparse variant of Macaulay's resultant matrix allows us to generalize the result of Subsection 3.1 to sparse affine equations. We consider here Laurent polynomials $f_1, \ldots, f_s$ in the the localization $R_{x_1\cdots x_n} = \mathbb{C}[x_1^{\pm 1} \ldots, x_n^{\pm 1}]$ of R at $x_1 \cdots x_n$. We want to find their roots in the algebraic torus $(\mathbb{C}^*)^n$. We assume that $I^* = \langle f_1, \ldots, f_s\rangle \cap R$ defines δ solutions, counting multiplicities. These are the solutions of I which are in $(\mathbb{C}^*)^n$.

Let V be a vector space of polynomials in R supported in some finite subset A of $\mathbb{N}^n$:

$$V = \bigoplus_{\alpha \in A} \mathbb{C} \cdot x^\alpha \subset R.$$

Let $W \subset V$ such that $x_i \cdot W \subset V, i = 1, \ldots, n$.

For the construction of N in the generic sparse square case we rely on the famous BKK-theorem by Bernstein [2], Kushnirenko [18] and Khovanskii [17] that bounds the number of solutions in the algebraic torus for a sparse square system.

This theorem states that given polytopes $P_1, \ldots, P_n \subset \mathbb{Z}^n$ a generic sparse square system $I = \langle f_1, \ldots, f_n\rangle \subset R_{x_1\cdots x_n}$ with supports $P_1, \ldots, P_n$ has as many solutions in $\mathbb{V}(I) \cap (\mathbb{C}^*)^n$ as the mixed volume $\mathrm{MV}(P_1, \ldots, P_n)$ of the polytopes P_i. The mixed volume $\mathrm{MV}(P_1, \ldots, P_n)$ of the n polytopes $P_1, \ldots, P_n$ is the coefficient of the monomial $\lambda_1\lambda_2\cdots\lambda_n$ in $\mathrm{Vol}_n(\sum_{i=1}^n \lambda_i P_i)$. More details can be found in [7, 15, 29] .

As in the dense generic case, we exploit the construction of sparse resultants from matrices to compute the truncated normal form N. Let f_0 be a generic linear polynomial and let $v \in \mathbb{R}^n$ be a generic, small n-tuple. We consider the resultant map

$$\begin{array}{rccc} \mathcal{R}_0 : & V_0 \times V_1 \times \cdots \times V_n & \longrightarrow & V \\ & (q_0, q_1, \ldots, q_n) & \longmapsto & q_0 f_0 + q_1 f_1 + \cdots + q_n f_n. \end{array}$$

where $V_i = \bigoplus_{\alpha\in A_i} \mathbb{C}\cdot x^\alpha$, $A_i = (P_0 + \ldots + \hat{P}_i + \ldots + P_n + v)\cap\mathbb{Z}^n$ (the notation $\hat{P}_i$ means that this term is left out) and $V = \bigoplus_{\alpha\in A} \mathbb{C}\cdot x^\alpha$, $A = (\sum_{i=0}^n P_i + v)\cap\mathbb{Z}^n$. We can select a square submatrix $\mathcal{R}'$ of this map, so that $\det(\mathcal{R}')$ is a nontrivial multiple of the toric resultant of $f_0, f_1, \ldots, f_n$ [12, 7]. We set $W = \{f \in V : x_i \cdot f \in V, i = 1, \ldots, n\}$ and $h = 1$. As for the Macaulay resultant matrix, we decompose $\mathcal{R}'$ in a block corresponding to the multiples of f_0 by a monomial set $\mathcal{B}_0 \subset W$ which is a basis of R/I and a block $\tilde{\mathcal{R}}$ for the multiples of $f_1, \ldots, f_n$. Then the null space N its of $\tilde{\mathcal{R}}^\top$ is such that $\ker(N) = \mathrm{im}(\tilde{\mathcal{R}}) = \mathrm{im}(\mathcal{R}) = I \cap V$. Since $\mathcal{B}_0 \subset W$, $N_{|W}$ is surjective and we can apply Theorem 2.2.

4. TNF from evaluations

In this section, we show how evaluations of the polynomials at well-chosen points can be used to construct the Truncated Normal Form N in polynomial bases which are not monomial bases. These polynomial bases are products of families of orthogonal polynomials in one variable.

We consider the dense affine case here but the approach can be extended to other families of systems. Recall that in this case $V = R_{\leq\rho}$, $W = R_{<\rho}$. Let $\{\phi_n(x)\}$ be a family of orthogonal univariate polynomials on an interval of $\mathbb{R}$, satisfying the recurrence relation $\phi_0(x) = 1$, $\phi_1(x) = a_0x + b_0$ and

$$\phi_{n+1}(x) = (a_nx + b_n)\phi_n(x) + c_n\phi_{n-1}(x)$$

with $b_n, c_n \in \mathbb{C}$, $a_n \in \mathbb{C}\backslash\{0\}$ so that $x\phi_n = \frac{1}{a_n}(\phi_{n+1} - b_n\phi_n - c_n\phi_{n-1}), n \geq 1$. For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, we define

$$\phi_\alpha(x) = \phi_\alpha(x_1, \ldots, x_n) = \prod_{i=1}^n \phi_{\alpha_i}(x_i).$$

We easily check that

$$x_i\phi_\alpha \quad = \quad \frac{1}{a_{\alpha_i}}(\phi_{\alpha+e_i} - b_{\alpha_i}\phi_\alpha - c_{\alpha_i}\phi_{\alpha-e_i})$$

with the convention that if $\beta \in \mathbb{Z}^n$ has a negative component, $\phi_\beta = 0$. We consider the basis $\mathcal{V} = \{\phi_\alpha : |\alpha| \leq \rho\}$ for V. The matrix $\mathcal{R}$ has columns indexed by all monomial multiples $x^\alpha f_i$ such that $x^\alpha f_i \in V$, and rows indexed by the basis $\mathcal{V}$. Let N be the left null space of $\mathcal{R}$. It represents linear functionals expressed in the dual of this basis. We have $\ker N \subset V \cap I$. Let $\mathcal{W} = \{\phi_\alpha : |\alpha| < \rho\}$ be a basis for W. The matrix $N_{|W} = N_{\mathcal{W}}$ is again a submatrix of columns indexed by $\mathcal{W}$. As before, we extract an invertible submatrix $N^* = N_{\mathcal{B}}$ from $N_{\mathcal{W}}$. If this is done using QR with pivoting, we have $\mathcal{B} = \{\phi_{\beta_1}, \ldots, \phi_{\beta_\delta}\}$ and $N^* = N_{\mathcal{B}}$ is the submatrix of $N_{\mathcal{W}}$ with columns indexed by $\mathcal{B}$. Let β_{ji} be the degree in x_i of ϕ_{β_j}. Then the j-th column of N_i is given by

$$(N_i)_j = \frac{1}{a_{\beta_{ji}}}(N_{\phi_{\beta_j+e_i}} - b_{\beta_{ji}}N_{\phi_{\beta_j}} - c_{\beta_{ji}}N_{\phi_{\beta_j-e_i}})$$

with the convention that an exponent with a negative component gives a zero column. Again, $M_{x_i} = (N^*)^{-1}N_i$ represents the multiplication by x_i in the basis $\mathcal{B}$ of R/I. The roots can then be deduced by eigen-computation as in the monomial case. Constructing the matrix $\mathcal{R}$ in this way can be done using merely function evaluations of the monomial multiples of the f_i by the properties of the orthogonal family $\{\phi_n\}$. This makes it particularly interesting to use bases for which there are fast ($O(d\log d)$) algorithms to convert a vector of function values to a vector of coefficients in the basis $\{\phi_n\}$. The Chebyshev basis is an example.

Truncated Normal Form in the Chebyshev basis. An interesting example of such a basis is the product Chebyshev bases. Recall that for the Chebyshev polynomials $\{T_n(x)\}$, the recurrence

relation is given by $a_0 = 1$, $a_n = 2, n > 0$, $b_n = 0, n \geq 0$, $c_n = -1, n > 0$. We get a basis $\mathcal{B} = \{T_{\beta_1}, \ldots, T_{\beta_\delta}\}$. In this basis we obtain

$$N_i = \frac{1}{2}(N_{\mathcal{B}_{+,i}} + N_{\mathcal{B}_{-,i}})$$

with $\mathcal{B}_{+,i} = \{T_{\beta_1+e_i}, \ldots, T_{\beta_\delta+e_i}\}$ and $\mathcal{B}_{-,i} = \{T_{\beta_1-e_i}, \ldots, T_{\beta_\delta-e_i}\}$ (negative exponents give a zero column by convention). Note that the expression is very simple here since the a_n, b_n, c_n are independent of n. Let $\omega_{k,d} = \cos\left(\frac{\pi(k+\frac{1}{2})}{d+1}\right)$, $k = 0, \ldots, d$. The decomposition of a polynomial $f(x)$ of degree d in the Chebyshev basis is $f = \sum_{i=0}^{d} c_i T_i$ where

$$c_j = \frac{2}{(d+1)} \sum_{k=0}^{d} f(\omega_{k,d})\, T_j(\omega_{k,d}).$$

For a polynomial $f(x) = f(x_1, \ldots, x_n) = \sum_\alpha c_\alpha T_\alpha(x)$ of degree d_i in x_i, we have

$$c_\alpha = \prod_i \frac{2}{(d_i+1)} \sum_{k_1=0}^{d_1} \cdots \sum_{k_n=0}^{d_n} f(\omega_{k,d})\, T_{\alpha_i}(\omega_{k,d})$$

with $\omega_{k,d} = (\omega_{k_1,d_1}, \ldots, \omega_{k_n,d_n})$. The coefficients c_α can be computed efficiently by taking a DCT of an array of function values of the monomial multiples of the f_i. The development of this technique is future research.

We conclude this section by noting that the monomials $\{x^n\}$ are a family of orthogonal polynomials on the complex unit circle and they satisfy the simple recurrence relation $x^{n+1} = x \cdot x^n$. This is an example of a so-called Szegő recurrence. Coefficients can be computed by taking a fast fourier transform of equidistant function evaluations on the unit circle. Such a Szegő recurrence exists for all families of orthogonal polynomials on the unit circle and hence products of these bases can also be used in this context [31].

5. Numerical experiments

In this section we show some numerical results. The aim is to show the potential of the proposed normal form algorithm as an alternative for some state of the art solvers. We develop a Matlab implementation of the algorithm to compute the multiplication tables, and the roots from those tables (step (2))[1]. For a description of how this second step works, see [5, 21, 11]. In a first subsection, we show how affine dense, affine sparse and homogeneous systems can be solved accurately using Algorithm 1. In Subsection 5.2 we summarize the comparison in [32] with the homotopy continuation packages PHCpack [34] and Bertini [1]. In Subsection 5.3 we compare Algorithm 1 to construct the multiplication matrices with a Groebner basis normal form method. We use Faugère's FGb [14] to compute a DRL Groebner basis of I and construct the multiplication matrices starting from this Groebner basis using the built in package Groebner of Maple. We use the classical monomial basis in the Matlab implementation of [32] for all the experiments. We have performed a few experiments using other bases, but we do not show them here for space reasons. In all of the experiments, the *residual* is a measure for the backward error computed as in [33]. Using double precision arithmetic, the best residual one can hope for is of order 10^{-16}.

5.1. **Some nontrivial examples.**

[1]An implementation in Julia has also been developed and is available at `https://gitlab.inria.fr/AlgebraicGeometricModeling/AlgebraicSolvers.jl`

d	δ	r	t (min)
50	2500	$5.55 \cdot 10^{-11}$	0.3
80	6400	$1.97 \cdot 10^{-10}$	4.9
100	10000	$1.31 \cdot 10^{-9}$	18
150	22500	$8.84 \cdot 10^{-9}$	184
160	25600	$3.85 \cdot 10^{-9}$	278
170	28900	$1.08 \cdot 10^{-7}$	370

TABLE 1. Numerical results for intersecting generic plane curves.

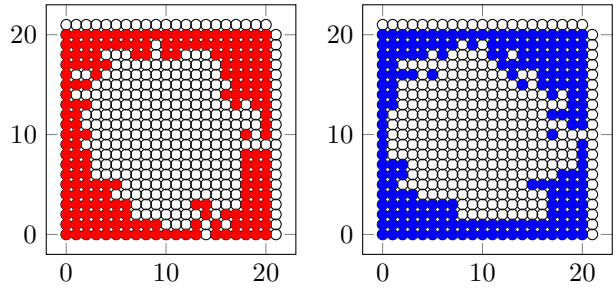

FIGURE 1. Monomials spanning V (○) and monomials in the basis for system 1 (●) and system 2 (●).

Intersecting two plane curves of degree 170. Consider all monomials of $\mathbb{C}[x_1, x_2]$ of degree $\leq d$ and assign a coefficient to each of these monomials drawn from a normal distribution with mean 0 and standard deviation 1. Doing this twice we obtain two dense polynomials $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$. These polynomials each define a curve of degree d in $\mathbb{C}^2$ and they consist of $\binom{2+d}{2}$ terms each ($\dim_{\mathbb{C}}(R_{\leq d}) = \binom{2+d}{2}$). The curves intersect in $\delta = d^2$ points, according to Bézout's theorem. To show the potential of our method, we have solved this problem for degrees up to 170 on a 128 GB RAM machine with a Xeon E5-2697 v3 CPU working at 2.60 GHz. This is the only experiment that was carried out with a more powerful machine. Table 1 shows some results. In the table, r gives an upper bound for the residual of all δ solutions and t is the total computation time in minutes. The number of terms in a degree $d = 170$ polynomial equals 14706. All of the following experiments are performed on an 8 GB RAM machine with an intel Core i7-6820HQ CPU working at 2.70 GHz, unless stated otherwise.

A sparse problem. We now consider $f_1, f_2 \in \mathbb{C}[x_1, x_2]$, each of bidegree (10,10). We construct two different systems. To every monomial in $\{x_1^{\alpha_1} x_2^{\alpha_2} : \alpha_1 \leq 10, \alpha_2 \leq 10\}$ we assign

(1) a coefficient drawn from a normal distribution with zero mean and $\sigma = 1$,
(2) a coefficient drawn from a (discrete) uniform distribution over the integers $-50, \ldots, 50$.

We refer to the resulting systems as system 1 and system 2 respectively. Algorithm 1 used as in Subsection 3.3 finds all 200 solutions with residual smaller than $1.43 \cdot 10^{-12}$ for system 1 and $8.01 \cdot 10^{-14}$ for system 2. Computations with polytopes are done using polymake [16]. We used QR with optimal column pivoting on $N_{|W}$ for the basis choice [33]. Figure 1 shows the resulting monomial bases for R/I for the two different systems, identifying in the usual way the monoid of monomials in two variables with $\mathbb{Z}^2$. Note that the basis does not correspond to a Groebner or border basis, it is not connected to 1. The total computation time was about 7 seconds for both systems.

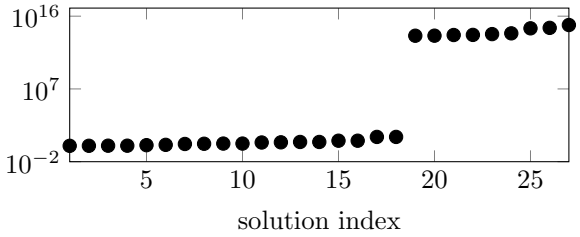

FIGURE 2. Norms of the computed solutions of 3 homogeneous equations in 4 variables in the affine chart $x_0 = 1$.

Solutions at infinity. Consider 3 dense homogeneous equations f_1, f_2, f_3 in $S = \mathbb{C}[x_0, \ldots, x_3]$ of degree 3 with normally distributed coefficients as before. According to Bézout's theorem, there are (with probability 1) 27 solutions in the affine chart $x_0 = 1$ of $\mathbb{P}^3$. We now manipulate the coefficients in the following way. Take the terms of f_2 not containing x_0 and replace the coefficients of f_1 standing with these monomials by the corresponding coefficients of f_2. Now f_1 and f_2 define the same curve of degree 3 in $\{x_0 = 0\} \simeq \mathbb{P}_2$ and this curve intersects with $f_3(0, x_1, x_2, x_3)$ in 9 points according to Bézout's theorem. Viewing $\{x_0 = 0\}$ as the hyperplane at infinity, we expect 9 solutions 'at infinity'. Numerically, the coordinate x_0 will be very small and we can detect solutions at infinity by sending the points in $\mathbb{P}^3$ to $\mathbb{C}^3$ by $(x_0 : x_1 : x_2 : x_3) \mapsto (x_1/x_0, x_2/x_0, x_3/x_0)$ and, for example, looking for points with large Euclidean norms. Figure 2 shows the norms of the computed solutions in this affine chart. There are indeed 9 solutions at infinity. The computation takes 0.02 seconds. Residuals are of order 10^{-12}. Doing the same for degree 10, 100 out of 1000 solutions lie at infinity. All solutions are found with residual no larger than $3.38 \cdot 10^{-11}$ within about 46 seconds.

5.2. **Comparison with homotopy solvers.** The homotopy continuation packages PHCpack and Bertini are standard tools for solving a system of polynomial equations [34, 1]. We define a *generic system* of degree d in n variables to be a system defined by n polynomials in $\mathbb{C}[x_1, \ldots, x_n]$ such that all polynomials have coefficients with all monomials of degree $\leq d$ drawn from a normal distribution with zero mean and $\sigma = 1$. From the numerical experiments in [32, 33] we learn that an advantage of algebraic methods over homotopy continuation methods is that they guarantee (assuming exact arithmetic) that *all* solutions are found. The homotopy packages (using standard settings) tend to give up on some of the paths once the systems become of larger degree and consistently miss out on some solutions. Table 2 illustrates this for $n = 2$ variables and degrees $d \geq 25$ (see tables in Subsection 8.5 of [32] for more details). In the table, r denotes the maximal residual of all computed solutions by the TNF algorithm, δ_{TNF} denotes the number of numerical solutions found by the TNF solver, Δ_S the number of solutions missed by the solver S and t_S is the computation time used by solver S to compute these δ_S solutions. Note that $\delta_{\text{TNF}} = d^2$ is the Bézout number. We used standard, double precision settings for the homotopy solvers in this experiment. The residual for the homotopy solvers is of order unit round-off since they work intrinsically with Newton refinement. A drawback of the method presented in this paper is that its complexity scales badly with the number of variables n. The involved matrices get much bigger than the number of solutions for larger n, which makes the nullspace computation very costly. Although Algorithm 1 is faster than both homotopy packages for $n = 2$ up to degree at least $d = 61$ (Table 2), for $n = 3$ the cross-over lies already at degree 8 or 9 and for $n = 5, d = 3$ the algebraic solver is already slower by a factor 20. One has to keep in mind that *all* solutions are found, though, with good accuracy. There are ways to reduce this complexity to make the algebraic method feasible for more variables and larger degrees, but this is future research.

d	r	δ_{TNF}	Δ_{phc}	Δ_{brt}	t_{TNF}	t_{phc}	t_{brt}
25	$1.21 \cdot 10^{-10}$	625	11	0	1.16	8.79	33.83
31	$5.23 \cdot 10^{-9}$	961	10	0	3.1	20.25	98.39
37	$4.05 \cdot 10^{-12}$	1,369	9	1	7.5	39.92	258.09
43	$1.74 \cdot 10^{-11}$	1,849	24	4	17.6	69.1	504.01
49	$1.57 \cdot 10^{-10}$	2,401	237	238	39.62	124.47	891.37
55	$1.84 \cdot 10^{-11}$	3,025	55	538	76.34	178.55	1,581.77
61	$3.26 \cdot 10^{-11}$	3,721	59	1,461	135.3	283.87	2,115.66

TABLE 2. Numerical results for PHCpack, Bertini and our method for dense systems in $n = 2$ variables of increasing degree d.

5.3. **Comparison with Groebner bases.** In this subsection we compare the proposed method with a Groebner basis normal form method. Once a monomial ordering is fixed, a reduced Groebner basis $g_1, \ldots, g_s$ provides a normal form onto the vector space B spanned by a set $\mathcal{B}$ of monomials, called a '*normal set*' [6]. This is the set of monomials that cannot be divided by any of the leading monomials of the polynomials in the Groebner basis. Any polynomial $f \in R$ can be written as

$$f = c_1 g_1 + \ldots + c_s g_s + r$$

with c_i and $r \in B$. Moreover, a Groebner basis has the property that such r is unique and the normal form is given by $\mathcal{N}(f) = r$ (it is easily checked that $\mathcal{N}$ is indeed a normal form). For the normal set $\mathcal{B}$ we denote $\mathcal{B} = \{x^{\beta_1}, \ldots, x^{\beta_\delta}\}$. The j-th column of the multiplication matrix M_{x_i} is then given by $\mathcal{N}(x^{\beta_j + e_i})$. This gives an algorithm for finding the multiplication operators M_{x_i}. Table 3 summarizes the steps of the algorithm and gives the corresponding steps of Algorithm 1.

We have used Faugère's FGb in Maple for step 1 [14] as a state of the art software for computing

	Our algorithm	GB algorithm
1	Construct $\mathcal{R}$ and compute N	Compute a DRL Groebner basis G which induces a normal form $\mathcal{N}$
2	QR with pivoting on $N_{\mid W}$ to find N^* corresponding to a basis $\mathcal{B}$ of R/I	Find a normal set $\mathcal{B}$ from G
3	Compute the N_i and set $M_{x_i} = (N^*)^{-1} N_i$	Compute the multiplication matrices by applying the induced normal form $\mathcal{N}$ on $x_i \cdot \mathcal{B}$

TABLE 3. Corresponding steps of our algorithm and the Groebner basis algorithm

Groebner bases. The routine `fgb_gbasis` computes a Groebner basis with respect to the degree reverse lexicographic (DRL) monomial order. For step 2, we used the command `NormalSet` from the built-in Maple package Groebner to compute a normal set from this Groebner basis. Step 3 is done using the command `MultiplicationMatrix` from the Groebner package.

An important note is that the Groebner basis computation has to be performed in exact arithmetic, because of its unstable behaviour. We will compare the speed of our algorithm with that of the Groebner basis algorithm for computing the matrices M_{x_i}. The multiplication operators computed by our algorithm correspond to another basis $\mathcal{B}$, as shown before, and they are computed in finite precision. We learn from the experiments that for the generic systems tested here, the resulting

operators give numerical solutions that are accurate up to unit round-off (in double precision) after one refining step of Newton's iteration. That is, the residues are never larger than order 10^{-9} and because of quadratic convergence the unit round-off ($\approx 10^{-16}$) is reached after one iteration. Using Maple, the multiplication matrices are found *exactly*, which is of course an advantage of the use of exact arithmetic. To compute the roots of the system, one can either reduce to computing exactly a minimal univariate polynomials from these operators of multiplication and solve it or compute the eigenvalues of these operators by numerical methods. This solving step is not integrated in the comparison.

The experiments will show that the use of floating point arithmetic leads to significantly less computation time, while the accuracy is still satisfactory for many applications.
We perform two different experiments: one in which the coefficients are floating point numbers up to 16 digits of accuracy that are converted in Maple to rational numbers, and one in which the coefficients are integers, uniformly distributed between -50 and 50. We restrict Matlab to the use of only one core since Maple also uses only one.

5.3.1. *Rational coefficients from floating point numbers.* We construct a generic system of degree d in n variables by assigning a coefficient to every monomial of degree $\leq d$ drawn from a normal distribution with mean zero and $\sigma = 1$ for each of the n polynomials defining the system. Computing the multiplication matrices via the implementation of Algorithm 1 in Matlab and the roots from their eigenstructure we observe that the residuals for the tested degrees are no larger than order 10^{-12}. We compare the computation time needed for finding the multiplication matrices using our algorithm with the time needed for the Groebner basis algorithm as described in Table 3. The float coefficients are approximated up to 16 digits of accuracy by a rational number in Maple, before starting the computation. This results in rational numbers with large numerators and denominators, which makes the computation in exact arithmetic very time consuming. Results are shown in Table 4. We conclude that the TNF method using floating point arithmetic can lead

n	d	t_{TNF}	t_{GB}	$t_{\text{GB}}/t_{\text{TNF}}$
2	2	$5.68 \cdot 10^{-4}$	$1.52 \cdot 10^{-2}$	26.76
2	3	$1.88 \cdot 10^{-3}$	$2.51 \cdot 10^{-2}$	13.34
2	4	$2.3 \cdot 10^{-3}$	$5.88 \cdot 10^{-2}$	25.57
2	5	$3.9 \cdot 10^{-3}$	0.19	47.96
2	6	$5.98 \cdot 10^{-3}$	0.48	79.55
2	7	$8.03 \cdot 10^{-3}$	1.16	143.89
2	8	$1.24 \cdot 10^{-2}$	2.85	229.04
2	9	$1.75 \cdot 10^{-2}$	6.19	354.39
2	10	$2.49 \cdot 10^{-2}$	14.27	573.24
3	2	$2.1 \cdot 10^{-3}$	$5.66 \cdot 10^{-2}$	27
3	3	$9.49 \cdot 10^{-3}$	1.82	191.54
3	4	$3.43 \cdot 10^{-2}$	52.19	1,520.51
3	5	0.12	893.38	7,186.04
4	2	$1.2 \cdot 10^{-2}$	1.31	109.76
4	3	0.27	910.96	3,391.25
5	2	0.15	59	398.27

TABLE 4. Timing results for the normal form algorithm of this paper (t_{TNF} (sec)) and the Groebner basis algorithm in Maple (t_{GB} (sec)) for generic systems in n variables of degree d with floating point coefficients drawn from a normal distribution with zero mean and $\sigma = 1$.

to a huge reduction of the computation time in these situations and, with the right choice of basis for the quotient algebra, the loss of accuracy is very small.

5.3.2. *Integer coefficients.* We now construct a generic system of degree d in n variables by assigning a coefficient to every monomial of degree $\leq d$ drawn from a discrete uniform distribution on the integers -50, ..., 50 for each of the n polynomials defining the system. Roots can be found using our algorithm with a residual no larger than order 10^{-10} for all the tested degrees. Table 5 shows that the Groebner basis method in exact precision is faster with these 'simple' coefficients, but the speed-up by using the TNF algorithm with floating point arithmetic is still significant.

n	d	t_{TNF}	t_{GB}	$t_{\mathrm{GB}}/t_{\mathrm{TNF}}$
2	2	$6.09 \cdot 10^{-4}$	$1.1 \cdot 10^{-2}$	18.06
2	4	$2.3 \cdot 10^{-3}$	$1.82 \cdot 10^{-2}$	7.91
2	6	$8.75 \cdot 10^{-3}$	$3 \cdot 10^{-2}$	3.43
2	8	$1.24 \cdot 10^{-2}$	$8.1 \cdot 10^{-2}$	6.51
2	10	$2.48 \cdot 10^{-2}$	0.15	5.88
2	12	$4.24 \cdot 10^{-2}$	0.38	8.89
2	14	$6.73 \cdot 10^{-2}$	0.71	10.56
2	16	0.1	1.32	12.62
2	18	0.16	2.33	14.91
2	20	0.2	4.31	21.42
2	22	0.29	7.07	24.64
2	24	0.5	11.55	23.09
2	26	0.62	19.36	31.08
2	28	0.81	29.25	36.22
2	30	1.08	41.01	37.89
3	2	$2.47 \cdot 10^{-3}$	$1.74 \cdot 10^{-2}$	7.05
3	3	$9.82 \cdot 10^{-3}$	$6.1 \cdot 10^{-2}$	6.21
3	4	$3.17 \cdot 10^{-2}$	0.33	10.4
3	5	$9.38 \cdot 10^{-2}$	2.09	22.33
3	6	0.27	10.42	38.67
3	7	1.31	45.4	34.62
3	8	5.3	168.03	31.72
3	9	16.16	573.45	35.5
3	10	41.71	1,674	40.14
4	2	$1.27 \cdot 10^{-2}$	$5.8 \cdot 10^{-2}$	4.58
4	3	0.18	3.19	17.86
4	4	8.89	99.78	11.23
4	5	145.36	2,367.04	16.28
5	2	$9.32 \cdot 10^{-2}$	0.4	4.28
5	3	73.16	286.15	3.91

TABLE 5. Timing results for the normal form algorithm of this paper (t_{TNF} (sec)) and the Groebner basis algorithm in Maple (t_{GB} (sec)) for generic systems in n variables of degree d with integer coefficients uniformly distributed between -50 and 50.

6. Conclusion and future work

We have used the results in [32] to propose new normal form constructions for computing the multiplication operators in the quotient algebra R/I associated to a zero-dimensional ideal. This leads to a fast and accurate algorithm for finding the roots of I, and all computations can be done in floating point arithmetic without loss of much accuracy. We have compared the method with homotopy and Groebner basis based solvers and the experiments show that for generic problems, the new method is competitive. The method finds a numerical approximation for *all* of the roots, under some genericity assumptions and can handle very challenging systems in low dimensions.

Making the method feasible for problems in higher dimensions and the implementation of truncated normal forms in different bases requires further research.

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. Bernstein. The number of roots of a system of equations. *Functional Anal. Appl.*, 9:1–4, 1975.

[3] L. Busé, M. Elkadi, and B. Mourrain. Resultant over the residual of a complete intersection. *Journal of Pure and Applied Algebra*, 164(1-2):35–57, Oct. 2001.

[4] 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.

[5] R. M. Corless, P. M. Gianni, and B. M. Trager. A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, pages 133–140. ACM, 1997.

[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] C. D'Andrea and M. Sombra. A Poisson formula for the sparse resultant. *Proceedings of the London Mathematical Society*, 110(4):932–964, Apr. 2015.

[9] P. Dreesen, K. Batselier, and B. De Moor. Back to the roots: Polynomial system solving, linear algebra, systems theory. *IFAC Proceedings Volumes*, 45(16):1203–1208, 2012.

[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] I. Z. Emiris and B. Mourrain. Matrices in Elimination Theory. *Journal of Symbolic Computation*, 28(1-2):3–44, 1999.

[14] J.-C. Faugère. FGb: A Library for Computing Groebner Bases. In K. Fukuda, J. Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.

[15] W. Fulton. *Introduction to toric varieties*. Number 131. Princeton University Press, 1993.

[16] M. Joswig, B. Müller, and A. Paffenholz. `polymake` and lattice polytopes. In *21st International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2009)*, Discrete Math. Theor. Comput. Sci. Proc., AK, pages 491–502. Assoc. Discrete Math. Theor. Comput. Sci., Nancy, 2009.

[17] A. Khovanskii. Newton polytopes and toric varieties. *Functional Anal. Appl.*, 11:289–298, 1977.

[18] A. Kushnirenko. Newton polytopes and the Bézout theorem. *Functional Anal. Appl.*, 10:233–235, 1976.

[19] F. S. Macaulay. Some formulae in elimination. *Proceedings of the London Mathematical Society*, 1(1):3–27, 1902.

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

[21] H. M. Möller and R. Tenberg. Multivariate polynomial system solving using intersections of eigenspaces. *Journal of symbolic computation*, 32(5):513–531, 2001.

[22] 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.

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

[24] B. Mourrain and J. P. Pavone. Subdivision methods for solving polynomial equations. *Journal of Symbolic Computation*, 44(3):292–306, 2009.

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

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

[27] L. Sorber, M. Van Barel, and L. De Lathauwer. Numerical solution of bivariate and polyanalytic polynomial systems. *SIAM J. Num. Anal. 52*, pages 1551–1572, 2014.

[28] H. J. Stetter. Matrix eigenproblems are at the heart of polynomial system solving. *ACM SIGSAM Bulletin*, 30(4):22–25, 1996.

[29] B. Sturmfels. Polynomial equations and convex polytopes. *The American Mathematical Monthly*, 105:907–922, 1998.
[30] B. Sturmfels. *Solving Systems of Polynomial Equations*. Number 97 in CBMS Regional Conferences. Amer. Math. Soc., 2002.
[31] G. Szegő. *Orthogonal Polynomials: 3d Ed*. American Mathematical Society, 1967.
[32] S. Telen, B. Mourrain, and M. Van Barel. Solving polynomial systems via a stabilized representation of quotient algebras. *arXiv preprint arXiv:1711.04543*, 2017.
[33] S. Telen and M. Van Barel. A stabilized normal form algorithm for generic systems of polynomial equations. *Preprint, arXiv:1708.07670*, 2017.
[34] 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.

Univ. Cote-d'Azur, Inria, Sophia-Antipolis, France
E-mail address: bernard.mourrain@inria.fr

KU Leuven, Belgium
E-mail address: simon.telen@cs.kuleuven.be

KU Leuven, Belgium
E-mail address: marc.vanbarel@cs.kuleuven.be