Truncated Normal Forms for Solving Polynomial Systems: Generalized and Efficient Algorithms

Bernard Mourrain, Simon Telen, Marc van Barel

Truncated Normal Forms for Solving Polynomial Systems: Generalized and Efficient Algorithms

Bernard Mourrain

AROMATH, INRIA, Sophia-Antipolis, 06902, France

Simon Telen

Department of Computer Science, KU Leuven, Heverlee, B-3001, Belgium

Marc Van Barel[1]

Department of Computer Science, KU Leuven, Heverlee, B-3001, Belgium

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. The framework presented in Telen et al. (2018) uses truncated normal forms (TNFs) to compute the algebra structure of R/I and the solutions of I. This framework allows for the use of much more general bases than the standard monomials for R/I. This is exploited in this paper to introduce the use of two special (non-monomial) types of basis functions with nice properties. This allows, for instance, to adapt the basis functions to the expected location of the roots of I. We also propose algorithms for efficient computation of TNFs and a generalization of the construction of TNFs in the case of non-generic zero-dimensional systems. The potential of the TNF method and usefulness of the new results are exposed by many experiments.

Keywords: polynomial systems, truncated normal forms, computational algebraic geometry, orthogonal polynomials

1. Introduction

Several problems in science and engineering boil down to the problem of finding the common roots of a set of multivariate (Laurent) 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 $I = \langle f_1, \ldots, f_s \rangle \subset R$ is the ideal generated by the polynomials $f_i \in R$, the problem can be formulated as finding the points in the algebraic set $\mathbb{V}(I) = \{z \in \mathbb{C}^n : f_i(z) = 0, i = 1, \ldots, s\} = \{z \in \mathbb{C}^n : f(z) = 0, \forall f \in I\}$. If $\mathbb{V}(I)$ is finite, I is called *zero-dimensional*. From now on, in this paper, I is a zero-dimensional ideal.

Email addresses: `bernard.mourrain@inria.fr` (Bernard Mourrain), `simon.telen@cs.kuleuven.be` (Simon Telen), `marc.vanbarel@cs.kuleuven.be` (Marc Van Barel)

The most important techniques for polynomial system solving are homotopy continuation methods (Bates et al. (2013); Verschelde (1999)), subdivision methods (Mourrain and Pavone (2009)) and algebraic methods (Emiris and Mourrain (1999); Sorber et al. (2014); Cox et al. (2006); Dreesen et al. (2012); Mourrain (1999); Stetter (1996)). See for instance Sturmfels (2002); Cattani et al. (2005) for an overview. Algebraic methods can be traced back to Bézout, Sylvester, Cayley, Macaulay.... Among them are *normal form methods*, which use rewriting techniques modulo I to turn the problem into an eigenvalue, eigenvector problem, see Cox et al. (2006); Elkadi and Mourrain (2007); Telen and Van Barel (2018); Telen et al. (2018). The key observation to translate the root finding problem into a linear algebra problem is a standard result in algebraic geometry: R/I is finitely generated over $\mathbb{C}$ as a $\mathbb{C}$-algebra (it is a finite dimensional $\mathbb{C}$-vector space with a compatible ring structure) 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 (Cox et al., 1992, Chapter 5, §3, Theorem 6). 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 $\mathbb{V}(I)$, see Elkadi and Mourrain (2007); Cox et al. (2006); Stetter (1996).

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.

We will now 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 $B = \text{span}(b_1, \ldots, b_\delta)$ along I. A well-known method to compute this projection map uses Groebner bases with respect to a certain monomial ordering (Cox et al. (1992, 2006)). The resulting basis consists of the monomials not in the initial of the ideal I. 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 are more flexible: they are not restricted to monomial orders. This makes border basis techniques more robust and potentially more efficient (Mourrain (1999, 2007); Mourrain and Trébuchet (2005, 2008)). In Telen and Van Barel (2018) it is shown that the choice of basis for R/I can be crucial for the accuracy of the computed multiplication maps and a 'heuristically optimal' monomial basis $\mathcal{B}$ is chosen using column pivoted QR factorization on a large Macaulay-type matrix for solving generic dense problems. Motivated by this, in Telen et al. (2018) a general algebraic framework is proposed for constructing so called *truncated normal forms* with respect to a numerically justified basis for R/I. By using the same QR operation to select $\mathcal{B}$, the resulting bases consist of monomials.

In this paper, we present an extension of the TNF method described in Telen and Van Barel (2018); Telen et al. (2018) to solve polynomial systems which are zero dimensional but not necessarily generic for a resultant construction. We describe techniques which reduce significantly the computational complexity of computing a TNF. We exploit the fact that the approach allows much more general constructions. We investigate the use of non-monomial bases to represent truncated normal forms. Although using monomial bases for R/I is standard, we argue that this is not always the most natural choice. For instance, if many of the points in $\mathbb{V}(I)$ are expected to be real, or we wish to know the real roots with high accuracy, Chebyshev polynomials prove to be good candidates. Another argument comes from the fact that a TNF on a finite dimensional vector space $V \subset R$ is a projector along $I \cap V$ onto $V/(I \cap V) \simeq R/I$. We show that replacing the column pivoted QR factorization in the algorithms of Telen and Van Barel (2018); Telen et al. (2018) by an SVD, the resulting TNF is in fact an orthogonal projection from the subspace $W = \{f \in V : x_i f \in V, i = 1, \dots, n\}$ onto R/I. The resulting basis $\mathscr{B}$ for R/I no longer consists of monomials in this case.

In the next section, we discuss TNFs and summarize some results from Telen et al. (2018) that are relevant for this work. In Section 3, we present a new algorithm for solving a non-generic system, using the TNF construction. In Section 4, we present methods to reduce the computational complexity of computing a TNF. In Section 5 we discuss TNFs constructed in non-monomial bases. We consider in particular bases obtained by using the SVD as an alternative for QR and orthogonal polynomial bases such as the Chebyshev basis for the construction of the resultant map. Finally, in Section 6, we show some experiments with the intention of illustrating the new results of this paper, but also of convincing the reader that the TNF algorithm is competitive with existing solvers in general.

2. Truncated normal forms

In this section, we briefly review the definitions and results from Telen et al. (2018) that are relevant for this paper. As in the introduction, denote $R = \mathbb{C}[x_1, \dots, x_n]$ and take an ideal $I = \langle f_1, \dots, f_s \rangle \subset R$ defining $\delta < \infty$ points, counting multiplicities. This is equivalent to the assumption that $\dim_{\mathbb{C}}(R/I) = \delta < \infty$. A *normal form* is a map characterized by the following properties.

Definition 1 (Normal form)**.** *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 the sequence*

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

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

From this definition it follows that multiplication with x_i in B is given by $M_{x_i} : B \to B : b \mapsto \mathcal{N}(x_i b)$. A truncated normal form is a restricted version of a normal form.

Definition 2 (Truncated normal form)**.** *Let $B \subset V \subset R$ with B, V finite dimensional vector subspaces, $x_i \cdot B \subset V, i = 1, \dots, n$ and $\dim_{\mathbb{C}}(B) = \delta = \dim_{\mathbb{C}}(R/I)$. A* Truncated Normal Form (TNF) *on V w.r.t. I is a linear map $\mathcal{N} : V \to B$ such that $\mathcal{N}$ is the restriction to V of a normal form w.r.t. I. That is, the sequence*

$$0 \longrightarrow I \cap V \longrightarrow V \xrightarrow{\mathcal{N}} B \longrightarrow 0$$

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

The constructions of TNFs proposed in Telen et al. (2018) work in two steps: a TNF is computed from a map $N : V \to \mathbb{C}^\delta$ which in turn is computed directly from the input equations using linear algebra techniques.

Definition 3. *If $\mathcal{N} : V \to B$ is a TNF and $N = P \circ \mathcal{N}$ for some isomorphism $P : B \to \mathbb{C}^\delta$, we say that N* covers *the TNF $\mathcal{N}$.*

If $N : V \to \mathbb{C}^\delta$ covers a TNF $\mathcal{N} : V \to B$, the above discussion suggests that $P = N_{|B}$ and $\mathcal{N} = (N_{|B})^{-1} \circ N = P^{-1} \circ N$ for some $B \subset V$ and that $M_{x_i} = (N_{|B})^{-1} \circ N_i = P^{-1} \circ N_i$ with $N_i = N_{|x_i \cdot B}$. The following is an immediate corrolary of Theorem 3.1 in Telen et al. (2018).

Theorem 4. *If $N : V \to \mathbb{C}^\delta$ covers a TNF on V w.r.t. I, then for any δ-dimensional $B \subset W = \{f \in V : x_i f \in V, i = 1, \ldots, n\}$ such that $N_{|B}$ is invertible, $\mathcal{N} = (N_{|B})^{-1} \circ N : V \to B$ is a TNF on V w.r.t. I.*

For a subpace $L \subset R$, denote $L^+ = \mathrm{span}_\mathbb{C}\{f, x_1 f, \ldots, x_n f \mid f \in L\} \subset R$. The vector space $W \subset V$ in the theorem is the maximal subspace $W \subset V$ such that $W^+ \subset V$. Theorem 4 leads to the following method for finding the δ roots of an ideal I (counted with multiplicity) from a TNF (see Telen et al. (2018) for more details).

```
$V \leftarrow$ a finite-dimensional subspace of $R$
$W \leftarrow$ the maximal subspace of $V$ such that $W^+ \subset V$
$N \leftarrow$ the matrix of a map $N : V \to \mathbb{C}^\delta$ such that $\ker N \subset I \cap V$
$N_{|W} \leftarrow$ columns of $N$ corresponding to the restriction of $N$ to $W$
**if** $N_{|W}$ surjective **then**
    $N_{|B} \leftarrow$ columns of $N_{|W}$ corresponding to an invertible submatrix
    $\mathcal{B} \leftarrow$ monomials corresponding to the columns of $N_{|B}$
    **for** $i = 1, \ldots, n$ **do**
        $N_i \leftarrow$ columns of $N$ corresponding to $x_i \cdot \mathcal{B}$
        $M_{x_i} \leftarrow (N_{|B})^{-1} N_i$
    **end for**
    $\Xi \leftarrow$ the roots of $I$ deduced from the tables of multiplication $M_{x_1}, \ldots, M_{x_n}$
**end if**
```

The construction of N can be based on resultant maps (see Section 4). It is shown in Telen et al. (2018), that for generic systems the cokernel map N of an appropriate resultant map covers a TNF.

3. Solving non-generic systems

In Telen et al. (2018) it is proved that a surjective map $N : V \to \mathbb{C}^\delta$ covers a TNF if and only if the following conditions are satisfied:

(a) $\exists u \in V$ such that $u + I$ is a unit in R/I,

(b) $\ker N \subset I \cap V$,

(c) $N_{|W} : W \to \mathbb{C}^\delta$ is surjective

with W as in Theorem 4. In general, it is fairly easy to construct a map N that satisfies the conditions (a) and (b) from a given set of generators of I (as the cokernel of a resultant map, see Section 4). In the generic case, it is known how to pick V and construct N such that also condition (c) is satisfied (see Telen et al. (2018)). In the non-generic case, where I is zero-dimensional but I defines (possibly infinitely many) points at infinity, this is more tricky. In this section we discuss how the roots of I may be retrieved in this situation from a map N that satisfies only conditions (a) and (b). In what follows, for a $\mathbb{C}$-vector space V we denote by V^* the dual space of linear forms from V to $\mathbb{C}$ and for $v \in V^*, f \in V$ we denote $\langle v, f\rangle = v(f)$ for the dual pairing. For a subspace $W \subset V$, let

$$W^\perp = \{v \in V^* \mid \forall f \in W, \langle v, f\rangle = 0\} \subset V^*.$$

For a matrix M, M^t denotes the transpose.

We consider a polynomial system given by $\mathbf{f} = (f_1, \ldots, f_s) \in R^s$ and the associated zero-dimensional ideal $I = \langle f_1, \ldots, f_s \rangle \subset R$. Suppose that we have a map $N : V \to \mathbb{C}^\delta$ satisfying conditions (a) and (b). Furthermore, suppose that $\dim_{\mathbb{C}}(R/I) = \delta' \leq \delta$ and I defines $r \leq \delta'$ distinct roots $\xi_1, \ldots, \xi_r \in \mathbb{C}^n$. We have $r = \delta'$ if and only if all the roots of I are simple. Given a basis $\mathcal{V}$ of V, a dual vector $v \in V^*$ is represented by a vector $(\langle v, b \rangle)_{b \in \mathcal{V}}$ in the dual basis of $\mathcal{V}$. If $A \subset \mathbb{N}^n$ is a finite subset of cardinality $\dim_{\mathbb{C}}(V)$ and V has a monomial basis $\mathcal{V} = \{x^\alpha\}_{\alpha \in A}$ (where $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$) then the vector representing an element $v \in V^*$ in the dual basis is $(\langle v, x^\alpha \rangle)_{\alpha \in A}$.

When R/I is of finite dimension δ' over $\mathbb{C}$, the vector space $I^\perp \subset R^*$ which can be identified with $(R/I)^*$, is of dimension δ' over $\mathbb{C}$. It contains the evaluations $\mathfrak{e}_{\xi_i} : f \mapsto f(\xi_i)$ at the roots ξ_i of I. These linear functionals are linearly independent since the points are distinct so that one can construct an associated interpolation polynomial family. We denote by $I^\perp_{|V}$ the restriction of elements of $I^\perp \subset R^*$ to V.

Let $J = \ker N$. By construction, $J \subset I \cap V$. The map $N : V \to \mathbb{C}^\delta$ is constructed from a basis of $J^\perp \subset V^*$. That is, each row of a matrix associated to N represents an element of V^*, which vanishes on J. As $J \subset I \cap V$, we have $I^\perp_{|V} \subset J^\perp$.

In order to recover the roots as eigenvalues, we will work with restrictions of N to subspaces of V. If $L \subset V$ is such a subspace, we denote $r_L = \dim_{\mathbb{C}}(\operatorname{im} N_{|L}) = \operatorname{rank}(N_{|L}) \leq \delta$. Let $W' \subset V' \subset V$ be subspaces satisfying

1. $\dim_{\mathbb{C}}(I^\perp_{|W'}) = \delta' = \dim_{\mathbb{C}}(R/I)$,
2. $(W')^+ \subset V'$,
3. $r_{W'} = r_{V'}$.

Note that the first condition is equivalent to saying that W' contains $w_1, \ldots, w_{\delta'}$ such that $\{w_1 + I, \ldots, w_{\delta'} + I\}$ is a basis for R/I. Because of condition 1 and $I^\perp_{|W'} \subset \operatorname{im} N^t_{|W'}$, we have a chain of inequalities $r \leq \delta' \leq r_{W'} = r_{V'} \leq \delta$. In what follows, with a slight abuse of notation, $N_{|L}$ is a matrix of the linear map $N_{|L}$ with respect to any basis of L. We are now ready to state the main result of this section.

Theorem 5. *Let $N : V \to \mathbb{C}^\delta$ be surjective with* $\ker N \subset I \cap V$. *Let $W' \subset V' \subset V$ satisfy conditions 1-3 above. Let $B' \subset W'$ such that* $\dim_{\mathbb{C}} B' = r_{B'} = r_{W'}$ *and let $N_0 = N_{|B'}$, $N_j = N_{|x_j \cdot B'}$. There are nonzero vectors $v_i \in \mathbb{C}^\delta \setminus \{0\}, i = 1, \ldots, r$ satisfying $N_j^t v_i = \xi_{i,j} N_0^t v_i$, where $\xi_{i,j}$ is the j-th coordinate of the root ξ_i of I, such that* $\operatorname{rank}(N_j^t - \xi_{i,j} N_0^t) < r_{B'}$.

Proof. As $\dim_{\mathbb{C}}(I^\perp) = \dim_{\mathbb{C}}(I^\perp_{|W}) = \delta'$ by the first condition, the restriction of the linear functionals $\mathfrak{e}_{\xi_i}$ to V' are linearly independent. Since $I^\perp_{|V'} \subset J^\perp = \operatorname{im}(N^t_{|V'})$, there exists an invertible matrix $U \in \mathrm{Gl}(\delta, \mathbb{C})$ such that the first r columns of $N^t_{|V'} U$ represent the evaluations $\mathfrak{e}_{\xi_1}, \ldots, \mathfrak{e}_{\xi_r}$ at the roots restricted to V' and the last but $r_{V'}$ columns of $N^t_{|V'} U$ are zero. That is, $N^t_{|V'} U$ looks like this:

$$N^t_{|V'} U = [\tilde{N}^t_{|V'} 0] = \begin{array}{c} \begin{matrix} \leftarrow & & r_{V'} & & \rightarrow & & & \end{matrix} \\ \begin{bmatrix} \vdots & \cdots & \vdots & \vdots & 0 & \cdots & 0 \\ (\mathfrak{e}_{\xi_1})_{|V'} & \cdots & (\mathfrak{e}_{\xi_r})_{|V'} & \vdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & 0 & \cdots & 0 \end{bmatrix} \end{array}.$$

Let $B' \subset W'$ such that $\dim_{\mathbb{C}} B' = r_{B'} = r_{W'}$. We have

$$N_0^t U = N^t_{|B'} U = [\tilde{N}_0^t \; 0], \qquad N_j^t = N^t_{|x_j \cdot B'} U = [\tilde{N}_j^t \; 0].$$

As $\dim_{\mathbb{C}} B' = r_{B'} = r_{W'} = r_{V'}$, the matrices $\tilde{N}_j^t \in \mathbb{C}^{r_{V'} \times r_{V'}}$, $j = 0, \ldots, n$ are square matrices and $\tilde{N}_0$ is invertible. Let $\mathcal{V}'$ be a basis of V' used to compute the matrix of N and let v_i be the i-th column of U. Note that $v_i \neq 0$ since $U \in \mathrm{Gl}(\delta, \mathbb{C})$. By construction, we have

$$N_{|V'}^t \, v_i = (\langle \mathfrak{e}_{\xi_i}, b \rangle)_{b \in \mathcal{V}'} = (b(\xi_i))_{b \in \mathcal{V}'}, \quad i = 1, \ldots, r.$$

Let $\mathcal{B}'$ be a basis of B' indexing the rows of $\tilde{N}_0^t$ and $\tilde{N}_j^t$. We have

$$N_0^t v_i = (b(\xi_i))_{b \in \mathcal{B}'}, \quad N_j^t v_i = (\xi_{i,j} b(\xi_i))_{b \in \mathcal{B}'} = \xi_{i,j} \, (b(\xi_i))_{b \in \mathcal{B}'}.$$

As $\tilde{N}_j^t - \xi_{i,j} \tilde{N}_0^t$ has a zero column, its rank is strictly less than $r_{B'}$. The theorem follows, since

$$\mathrm{rank}(N_j^t - \xi_{i,j} N_0^t) = \mathrm{rank}((N_j^t - \xi_{i,j} N_0^t) U) = \mathrm{rank}(\tilde{N}_j^t - \xi_{i,j} \tilde{N}_0^t).$$

□

Let $U \in \mathrm{Gl}(\delta, \mathbb{C})$ be any matrix such that the last but $r_{V'}$ columns of $N_{|V'}^t U$ are zero and let $B, N_j, j = 0, \ldots, n$ be as in Theorem 5. By the theorem, the square pencils $\tilde{N}_j^t - \lambda \tilde{N}_0^t$, where $\tilde{N}_j^t$ contains the first $r_{W'} = r_{V'}$ columns of $N_j^t U$, are regular and among their eigenvalues are $\lambda = \xi_{i,j}$. Such a matrix U is for instance obtained from a QR factorization[3] with column pivoting of $N_{|W'}$: $N_{|W'} \mathbf{P} = \mathbf{QR}$. This leads at the same time to a basis $\mathcal{B}'$ corresponding to the monomials selected by the first $r_{W'} = r_{V'}$ columns of $\mathbf{P}$. We will show in Section 5 that alternatively, the singular value decomposition can be used. This leads to Algorithm 1 for finding the roots of I.

Algorithm 1 Computes the roots of a non-generic system

```
1: procedure SolveNonGeneric($f_1, \ldots, f_s$)
2:     Res ← a resultant map from $V_1 \times \cdots \times V_s$ to $V$
3:     $N$ ← cokernel map of Res
4:     $W', V'$ ← Subspaces of $V$ satisfying conditions 1-3
5:     $\mathbf{Q}, \mathbf{R}, \mathbf{P}$ ← QR-factorization with pivoting of $N_{|W'}$
6:     $\mathcal{B}'$ ← monomials corresponding to the first $r_{V'}$ columns of $\mathbf{R}$
7:     $\tilde{N}_0$ ← first $r_{V'}$ rows and columns of $\mathbf{R}$
8:     for $i = 1, \ldots, n$ do
9:         $\tilde{N}_i$ ← first $r_{V'}$ rows of the submatrix of $\mathbf{R}$ with columns indexed by $x_i \cdot \mathcal{B}'$
10:    end for
11:    $\Xi$ ← roots of $I = \langle f_1, \ldots, f_s \rangle$ computed as eigenvalues from $(\tilde{N}_0, \ldots, \tilde{N}_n)$
12:    return $\Xi$
13: end procedure
```

The common eigenvectors of the pencils $\tilde{N}_j^t - \lambda \tilde{N}_0^t$ can be obtained by computing the generalized eigenvectors of a random combination of $\tilde{N}_1^t, \ldots, \tilde{N}_n^t$ and $\tilde{N}_0^t$, and by selecting those which are common to the pencils $\tilde{N}_j^t - \lambda \tilde{N}_0^t$ for $i = 1, \ldots, n$. Since $\tilde{N}_0$ is invertible, the pencils are *regular*. A standard QZ algorithm can be used to find the eigenpairs. To conclude this section, we briefly discuss some important cases in which Theorem 5 can be used. In what follows, $R_{\leq d}$ is the vector subspace of polynomials of degree at most d.

[3]We use bold capital letters for factor matrices in standard factorizations in linear algebra, so that we can use the usual letters (e.g. **R** in QR, **V** in SVD, ...) without being inconsistent with the notation of this paper (e.g. $R, V, \ldots$).

- In the generic case one can take $V' = V$ and $W' = W$ from the standard construction and $(\tilde{N}_0^t)^{-1}\tilde{N}_j^t$ is a matrix of the multiplication map M_{x_j} (see Telen et al. (2018) for more details).

- In the case of finitely many solutions in projective space, let ρ be the smallest number such that $\dim_{\mathbb{C}}(R_{\leq\rho+k}/I_{\leq\rho+k}) = \delta'$ for $k \geq 0$ (ρ is the *degree of regularity* in the affine sense). Taking $V = R_{\leq\rho+2}, V' = R_{\leq\rho+1}, W' = R_{\leq\rho}$ gives a regular pencil with only finite eigenvalues corresponding to the solutions ξ_i. Note that in this case $r_{V'} = \delta'$ and $N_{|V'} : V' \to \mathbb{C}^{\delta'}$ covers a TNF. It follows again that the $(\tilde{N}_0^t)^{-1}\tilde{N}_j^t$ are multiplication matrices. An alternative in this case is to use a random linear change of coordinates to apply the generic TNF construction, or to use Algorithm 3 in Telen et al. (2018), which computes also the points at infinity in their homogeneous coordinates.

- If there are positive dimensional solution sets at infinity, a ρ sufficiently large as in the previous bullet also exists (For instance *rho* larger than the degree of the relations describing a Grobner basis for the graded reverse lexicographic ordering in terms of the polynomials f_i). An example is given in Subsection 6.4.

- Note that if N is constructed from a resultant map with respect to an ideal $J \subset I$, then $\ker N \subset J \cap V \subset I \cap V$. This means that Algorithm 1 can be used to find the isolated roots of ideals defining varieties with positive dimensional irreducible components. An example is given in Subsection 6.4.

4. Efficient construction of TNFs

An important step in the TNF method for solving polynomial systems is the computation of a map N that covers a TNF. In some important cases, such a map can be obtained from a resultant map. We start this section with a brief description of how that works.

Definition 6 (Resultant map)**.** *Let* $\mathbf{f} = (f_1, \ldots, f_s) \in R^s$. *A* resultant map *w.r.t.* $\mathbf{f}$ *is a map*

$$\text{Res} : V_1 \times \cdots \times V_s \longrightarrow V : \quad (q_1, \ldots, q_s) \longmapsto q_1 f_1 + \cdots + q_s f_s.$$

with $V_i, V \subset R$ *finite dimensional vector subspaces.*

The *cokernel* (N, C) of a linear map Res consists of a linear map $N : V \to C$ and a $\mathbb{C}$-vector space C, unique up to isomorphism, such that $N \circ \text{Res} = 0$ and any linear map $N' : V \to C'$ satisfying $N' \circ \text{Res} = 0$ factors through N. Clearly, $C \simeq V/(\text{im Res})$ and $N : V \to V/(\text{im Res})$ is the straightforward projection. In what follows, with a slight abuse of notation, by the cokernel of Res we mean the map N. In Telen et al. (2018) it is shown how the cokernel of a specific resultant map covers a TNF in the following important cases.

1. When $s = n$ and the equations are dense of degree $d_i = \deg(f_i)$, N is the cokernel of the resultant map defined by

$$V_i = R_{\leq \sum_{j \neq i}(d_j - 1)}, \; V = R_{\leq \sum_{i=1}^n d_i - (n-1)}.$$

2. If $s = n$ and the equations are sparse and generic with respect to their Newton polytopes, N is the cokernel of the resultant map defined as follows. Denote $P_i \subset \mathbb{R}^n$ for the Newton polytope of f_i and let v be a generic small real n-vector.

$$V_i = \bigoplus_{\alpha \in A_i} \mathbb{C} \cdot x^\alpha, \qquad V = \bigoplus_{\alpha \in A} \mathbb{C} \cdot x^\alpha$$

with $A_i = (P_1 + \ldots + \hat{P}_i + \ldots + P_n + \Delta_n + v) \cap \mathbb{Z}^n$ ($\hat{\cdot}$ means this term is left out of the sum), $A = (P_1 + \ldots + P_n + \Delta_n + v) \cap \mathbb{Z}^n$ and Δ_n the standard simplex.

Similar constructions can be used to solve complete intersection in projective space or Segre varieties, and there are ways to deal with the case $s > n$ as well. We recall from Section 3 that in some cases the cokernel of a resultant map does not cover a TNF but it can still be used to compute the roots of I.

The TNF method for solving polynomial systems, like other algebraic approaches, has the important drawback that the complexity scales badly with the number n of variables. This is due to the fact that the complexity of computing the cokernel map of the appropriate resultant map increases drastically with n. We describe now two possible techniques to reduce this drastic increase of complexity. The first one computes the cokernel map degree by degree. This technique has also been exploited in Batselier et al. (2014). The second one exploits the redundancy in the vector spaces V_i in the definition of the resultant map.

4.1. Computing the cokernel degree by degree

We consider the case where $V = R_{\leq\rho}$ for some degree ρ (of regularity). For instance, the square dense generic case or the overdetermined case with finitely many solutions in projective space. Let I be generated by $f_1, \ldots, f_s$ with $d_i = \deg(f_i)$. We define the resultant maps

$$\mathrm{Res}_k : V_{1,k} \times \cdots \times V_{s,k} \to V_k, \quad k = 1, \ldots, \rho$$

such that $V_k = R_{\leq k}$, $V_{i,k} = R_{\leq k-d_i}$ with the convention that $R_{\leq k} = \{0\}$ when $k < 0$. Let $N_k : V_k \to \mathbb{C}^{\delta_k}$ be the cokernel of Res_k. We have that $\mathrm{Res}_\rho = \mathrm{Res}$ and $N_\rho = N$ is the map we want to compute. Our aim here is to compute N_{k+1} from N_k in an efficient way. Note that $V_k \subset V_{k+1}$, $V_{i,k} \subset V_{i,k+1}$. We write

$$\mathrm{Res}_{k+1} : S_k \times T_{k+1} \to V_{k+1}$$

where $S_k = V_{1,k} \times \cdots \times V_{s,k}$, $T_{k+1} \simeq \prod_{i=1}^{s} V_{i,k+1}/V_{i,k}$ and $(\mathrm{Res}_{k+1})_{|S_k} = \mathrm{Res}_k$. Define

$$H_{k+1} = V_{k+1}/V_k, \quad \hat{N}_{k+1} : V_k \times H_{k+1} \to \mathbb{C}^{\delta_k} \times H_{k+1} : (v, w) \mapsto (N_k(v), w).$$

Furthermore, set $\mathrm{Res}'_{k+1} = (\mathrm{Res}_{k+1})_{|T_{k+1}}$. Here is what the matrices look like:

$$\hat{N}_{k+1} = \begin{array}{c|c|c} & V_k & H_{k+1} \\ \hline \mathbb{C}^{\delta_k} & N_k & 0 \\ \hline H_{k+1} & 0 & \mathrm{id}_{H_{k+1}} \end{array}, \qquad \mathrm{Res}_{k+1} = \begin{array}{c|c|c} & S_k & T_{k+1} \\ \hline V_k & \mathrm{Res}_k & A_{k+1} \\ \hline H_{k+1} & 0 & B_{k+1} \end{array}.$$

Finally, define $L_{k+1} : \mathbb{C}^{\delta_k} \times T_{k+1} \to \mathbb{C}^{\delta_{k+1}}$ as the cokernel of $\hat{N}_{k+1} \circ \mathrm{Res}'_{k+1}$ where $\mathrm{Res}'_{k+1} = \begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix}$.

Theorem 7. *The map $N_{k+1} = L_{k+1} \circ \hat{N}_{k+1}$ is the cokernel of* Res_{k+1}, *i.e.*

$$\prod_{i=1}^{s} V_{i,k+1} \xrightarrow{\mathrm{Res}_{k+1}} V_{k+1} \xrightarrow{L_{k+1} \circ \hat{N}_{k+1}} \mathbb{C}^{\delta_{k+1}} \longrightarrow 0$$

is exact.

Proof. By definition $\sigma \in \operatorname{coker}\mathrm{Res}_k \Leftrightarrow \mathrm{Res}_k^t\sigma = 0 \Leftrightarrow \sigma = N_k^t(\omega)$ for some $\omega \in \mathbb{C}^{\delta_k}$. Similarly $(\sigma,\tau) \in V_k^* \oplus H_{k+1}^* = V_{k+1}^*$ is in $\operatorname{coker}\mathrm{Res}_{k+1}$ if and only if

$$\begin{cases} \mathrm{Res}_k^t\sigma = 0, \quad \text{i.e. } \sigma = N_k^t(\omega), \\ A_{k+1}^t\sigma + B_{k+1}^t\tau = 0 = A_{k+1}^t N_k^t\omega + B_{k+1}\tau. \end{cases}$$

Equivalently, $(\sigma,\tau) = \hat{N}_{k+1}^t(\omega,\tau)$ with (ω,τ) in the cokernel L_{k+1} of $\hat{N}_{k+1} \circ \mathrm{Res}'_{k+1}$. The theorem follows. □

This means that if we have computed N_k, then we can compute N_{k+1} by computing the cokernel L_{k+1} of $\hat{N}_{k+1} \circ \mathrm{Res}'_{k+1}$ instead of Res_{k+1}. This reduces the computational complexity significantly for $n > 2$. We show some results in Subsection 6.5.

4.2. Reducing the size of Res

As explained above, a map N covering a TNF is usually computed as the cokernel of a resultant map

$$\mathrm{Res} : V_1 \times \cdots \times V_n \to V.$$

The vector spaces V_i and V depend on the input equations. The definitions of the V_i at the beginning of this section for the generic, square case are derived from the Macaulay and toric resultant matrix constructions (Telen et al. (2018); Cox et al. (2006); Emiris and Mourrain (1999)) and the close relation of Res to resultant matrices is the reason Res is called a resultant map. (In resultant constructions an additional polynomial f_0 is usually involved.)

In these resultant maps based on Macaulay and toric resultant constructions, there is a proper subspace of $V_1 \times \cdots \times V_n$ such that if we restrict Res to this subspace, it has the same image. Therefore Res is column rank deficient. However, in the generic case, we know that the rank of Res is $l - \delta$ where $l = \dim_{\mathbb{C}}(V)$. This means that taking $l - \delta$ random linear combinations of the columns of Res gives a matrix with the same rank and the same cokernel. This comes down to restricting Res to a random linear subspace of $V_1 \times \cdots \times V_n$, instead of the very specific one from the resultant matrix constructions (see (Cox et al., 2006, Chapter 3)). We may hope that this procedure results in better numerical behaviour, and the experiments in Subsection 6.5 show that it does. Let us denote $l_i = \dim_{\mathbb{C}}(V_i)$. By restricting to a random subspace of the right dimension, we reduce the number of columns of Res from $l_1 + \ldots + l_n$ to $l - \delta$. To summarize: instead of computing the cokernel of $\mathrm{Res} \in \mathbb{C}^{l \times (l_1 + \ldots + l_n)}$, we compute the cokernel of the product $\mathrm{Res}C \in \mathbb{C}^{l \times (l-\delta)}$ where $C \in \mathbb{C}^{(l_1+\ldots+l_n)\times(l-\delta)}$ is a matrix with random entries (for instance, real and drawn from a normal distribution with zero mean and $\sigma = 1$).

Example 8. *In the case of a dense, square system defined by n generic equations in n variables, each of degree d, we have*

$$l_i = \binom{(n-1)d+1}{(n-1)(d-1)}, i = 1,\ldots,n, \quad l = \binom{nd+1}{n(d-1)+1}, \quad \delta = d^n$$

where $\binom{h}{k} = \frac{h!}{k!(h-k)!}$. *The reduction in the number of columns is illustrated in Figure 1.*

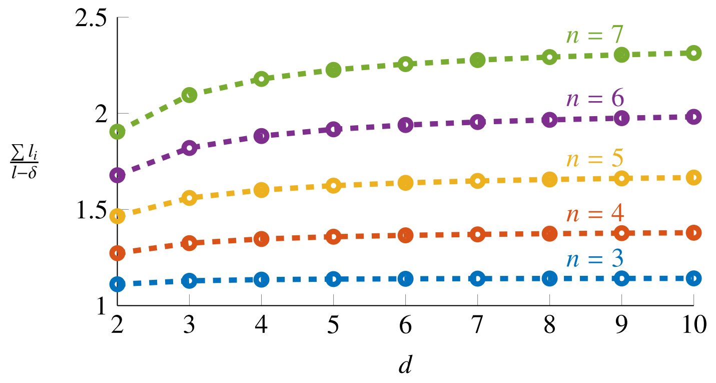


Figure 1: The ratio $(l_1 + \ldots + l_n)/(l - \delta)$ of the number of columns of Res and Res$C$ for increasing values of $n = 3, 4, 5, 6, 7$ and degrees $d = 2, \ldots, 10$, in the context of Example 8.

5. TNFs in non-monomial bases

In this section, we deal with matrix representations of the linear maps from Section 2: we fix bases for the involved vector spaces. For $\mathbb{C}^\delta$, we will use the canonical basis $\{e_1, \ldots, e_\delta\}$. We denote $\mathcal{V} = \{v_1, \ldots, v_l\} \subset V$ for a basis of V ($l = \dim_{\mathbb{C}}(V)$) and $\mathcal{W} = \{w_1, \ldots, w_m\} \subset W$, $m < l$ for a basis of $W = \{f \in V : x_i f \in V, i = 1, \ldots, n\}$. Analogously, $\mathcal{B} = \{b_1, \ldots, b_\delta\}$ is a basis for B. For simplicity, we assume $\mathcal{W} \subset \mathcal{V}$. To simplify the notation we will make no distinction between a matrix and the abstract linear map it represents.

Suppose we have a map $N : V \to \mathbb{C}^\delta$ which covers a TNF $\mathcal{N} : V \to B$ for some $B \subset W \subset V$. In practice, this means that we have a matrix representation of N with respect to a fixed basis $\mathcal{V}$ of V. Since N is usually computed as the cokernel of a resultant map Res, using for instance the SVD, the basis $\mathcal{V}$ is usually induced by the basis used for V to represent Res. Note that since we are assuming $\mathcal{W} \subset \mathcal{V}$, $N_{|W} : W \to \mathbb{C}^\delta$ is just a $\delta \times m$ submatrix of N consisting of the columns indexed by $\mathcal{W}$. In this case we write $N_{\mathcal{W}} = N_{|W}$. To recover $\mathcal{N}$ from N, all that is left to do is compute the matrix $N_{|B} : B \to \mathbb{C}^\delta$ with respect to a fixed basis $\mathcal{B} = \{b_1, \ldots, b_\delta\}$ of $B \subset W$. Then the matrix of $\mathcal{N}$ with respect to the bases $\mathcal{V}$ for V and $\mathcal{B}$ for B is $\mathcal{N} = (N_{|B})^{-1} N$. Note that if $\mathcal{B} \subset \mathcal{W}$, the matrix $N_{\mathcal{B}} = N_{|B}$ consists of a subset of δ columns of $N_{|W}$. Since $B \subset R$ is identified with R/I in the TNF framework, the set $\mathcal{B}$ of basis elements represents a basis for $\{b_1 + I, \ldots, b_\delta + I\}$ of R/I. Traditionally, e.g. in resultant and Groebner basis contexts, but often for border bases as well, the b_i are monomials. In this section, we step away from this and show that it is sometimes natural to use non-monomial bases. The following three scenarios clearly lead to non-monomial bases of R/I.

1. The set $\mathcal{V}$ consists of monomials, but $B \subset W$ is computed using another procedure, such that $\mathcal{B} \not\subset \mathcal{W}$. An example is discussed in the first subsection, where we use a SVD of $N_{\mathcal{W}}$ to select $\mathcal{B}$ instead of a QR decomposition.

2. The set $\mathcal{V}$ consists of non-monomial basis elements of V and $\mathcal{B} \subset \mathcal{W} \subset \mathcal{V}$. This happens, for instance, when $\mathcal{B}$ is chosen by performing a QR with optimal column pivoting on the matrix $N_{\mathcal{W}}$. The column pivoting comes down to a pivoting of the elements in $\mathcal{W}$, and $N_{|B}$ is simply a $\delta \times \delta$ submatrix $N_{\mathcal{B}}$ of $N_{\mathcal{W}}$. This situation is discussed in the second subsection for a specific type of basis functions.

3. It is straightforward to combine these first two scenarios, such that $\mathcal{V}$ does not contain (only) monomials and $\mathcal{B} \not\subset \mathcal{W}$.

5.1. TNFs as orthogonal projectors

In the approach described in Telen et al. (2018), the selection of a basis $\mathcal{B}$ (see Section 2) is performed through a column pivoted QR factorization of $N_{|W}$. We present an alternative basis selection using the singular value decomposition,which is another important tool from numerical linear algebra (Trefethen and Bau III (1997)). This provides a basis $\mathcal{B}$, which is not a monomial basis. Let $\mathcal{V} = \{x^\alpha : \alpha \in A\}$ be a set of monomials corresponding to a finite set $A \subset \mathbb{Z}^n$ of lattice points such that $\mathcal{W} = \{x^{\alpha_1}, \ldots, x^{\alpha_m}\} \subset \mathcal{V}$ is a basis of W. We decompose

$$N_{\mathcal{W}} = \mathbf{U}\mathbf{S}\mathbf{V}^H$$

with $\cdot^H$ the Hermitian transpose. We split $\mathbf{S}$ and $\mathbf{V}$ into compatibly sized block columns:

$$N_{\mathcal{W}}\,[\mathbf{V}_1\ \mathbf{V}_2] = \mathbf{U}\,[\hat{\mathbf{S}}\ \ 0]$$

with $\hat{\mathbf{S}}$ diagonal and invertible ($N_{|W}$ is onto). In analogy with the QR case, we take

$$\mathcal{B} = [x^{\alpha_1}\ \cdots\ x^{\alpha_m}]\,\mathbf{V}_1,$$

such that $B = \operatorname{span}(\mathcal{B}) \simeq \operatorname{im} \mathbf{V}_1$. Therefore

$$(N_{\mathcal{W}})_{|B} = N_{|B} = \mathbf{U}\,[\hat{\mathbf{S}}\ \ 0]\,[\mathbf{V}_1\ \mathbf{V}_2]^H\,\mathbf{V}_1 = \mathbf{U}\hat{\mathbf{S}}.$$

This tells us that the singular values of $N_{|B}$ are the singular values of $N_{\mathcal{W}}$ and $\mathcal{N}_{|W} = (N_{|B})^{-1}N_{\mathcal{W}} = \mathbf{V}_1^H$. Since $\ker N_{\mathcal{W}} = I \cap W \simeq \operatorname{im} \mathbf{V}_2 \subset \mathbb{C}^m$ and $\operatorname{im} \mathbf{V}_1 \perp \operatorname{im} \mathbf{V}_2$ by the properties of the SVD, we see that

$$(I \cap W) \perp B$$

with respect to the standard inner product in $\mathbb{C}^m$ and using coordinates w.r.t $\mathcal{W}$. Equivalently, with this choice of B, $\mathcal{N}_{|W} = \mathbf{V}_1^H$ projects W orthogonally onto B. The obtained basis $\mathcal{B}$ is an orthonormal basis for the orthogonal complement B of $I \cap W$ in W. This makes B somehow a unique 'canonical' representation of R/I w.r.t. $\mathcal{W}$. Orthogonality is a favorable property for a projector, because the sensitivity of the image to perturbations of the input is minimal. Also, since $\mathcal{N}_{|W}(f) \perp (I \cap W), \forall f \in W$, $\|\mathcal{N}_{|W}(f)\|$ is a natural measure for the *distance* of f to the ideal in the basis $\mathcal{W}$, which is induced by the Euclidean distance in $\mathbb{C}^m$. We note that $\mathcal{N}$ does not project V orthogonally onto B. In order to have an orthogonal projector $\mathcal{N}_{|W'} : W' \to B$, one must take V large enough such that $W' \subset W \subset V$. Following this procedure, $\mathcal{B}$ is a non-monomial basis of B (or R/I) consisting of δ polynomials supported in $\mathcal{W}$. The above discussion shows that in some sense, $\mathcal{B}$ gives a 'natural' basis for R/I, given the freedom of choice provided by Theorem 4. Unlike the QR algorithm, there are no heuristics involved. For the root finding problem, we observe that $\mathcal{B}_{\mathrm{SVD}}$ has the same good numerical properties as $\mathcal{B}_{\mathrm{QR}}$. We show some numerical examples in Section 6.

5.2. TNFs from function values

We consider the dense square case ($n = s$) here but the approach can be extended to other families of systems. Recall that in this case $V = R_{\leq\rho}$, $W = R_{<\rho}$ where $\rho = \sum_{i=1}^n d_i - (n-1)$.

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

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

where $e_i \in \mathbb{Z}^n$ is a vector with all zero entries except for a 1 in the i-th postion and 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 with $|\alpha| = \sum_{i=1}^n \alpha_i$. The matrix Res can be constructed such that it has columns indexed by all monomial multiples $x^\alpha f_i$ such that $x^\alpha f_i \in V$ (we use monomial bases for the V_i), and rows indexed by the basis $\mathcal{V}$. The corresponding cokernel matrix represents a map $N : V \to \mathbb{C}^\delta$ covering a TNF. The set $\mathcal{W} = \{\phi_\alpha : |\alpha| < \rho\} \subset \mathcal{V}$ is a basis for W. The matrix $N_{|W} = N_{\mathcal{W}}$ is again a submatrix of columns indexed by $\mathcal{W}$. To compute a TNF, we have to compute an invertible matrix $N_{|B}$ from $N_{\mathcal{W}}$. If this is done using QR with pivoting, we have $\mathcal{B} = \{\phi_{\beta_1}, \ldots, \phi_{\beta_\delta}\} \subset \mathcal{W}$ and $N_{|B} = N_{\mathcal{B}}$ 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 $N_i = N_{|x_i \cdot B}$ 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. Recall from Section 2 that $M_{x_i} = (N_{|B})^{-1}N_i$ 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 Res 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 $\{\phi_n\}$. We now discuss the Chebyshev basis as an important example.

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. We define

$$\omega_{k,d} = \cos\left(\frac{\pi(k+\frac{1}{2})}{d+1}\right), \quad k = 0, \ldots, d.$$

Let $f = \sum_{j=0}^d c_j T_j$ be the representation in the Chebyshev basis of a polynomial $f \in \mathbb{C}[x]$ and define $f_k = f(\omega_{k,d})$. By the property of T_i that $T_n(x) = \cos(n \arccos(x))$ for $x \in [-1, 1]$, we have

$$f_k = \sum_{j=0}^{d} c_j \cos\left(\frac{j\pi(k+\frac{1}{2})}{d+1}\right). \tag{1}$$

Comparing (1) to the definition of the (type III) discrete cosine transform (DCT) $(Z_k)_{k=0}^d$ of a sequence $(z_k)_{k=0}^d$ of $d+1$ complex numbers[4]

$$Z_k = \sqrt{\frac{2}{d+1}}\left(\frac{1}{\sqrt{2}} z_0 + \sum_{j=1}^{d} z_j \cos\left(\frac{j\pi(k+\frac{1}{2})}{d+1}\right)\right),$$

we see that

$$\sqrt{\frac{2}{d+1}}(f_k)_{k=0}^d = \mathrm{DCT}\left((\sqrt{2}c_0, c_1, \ldots, c_d)\right).$$

We conclude that the coefficients c_k in the Chebyshev expansion can be computed from the function evaluations f_k via the inverse discrete cosine transform (IDCT), which is the DCT of type II:

$$z_k = \sqrt{\frac{2}{d+1}}\left(\sum_{j=0}^{d} Z_j \cos\left(\frac{k\pi(j+\frac{1}{2})}{d+1}\right)\right).$$

This gives

$$c_k = \left(\frac{1}{\sqrt{2}}\right)^{q_k}\left(\sqrt{\frac{2}{d+1}}\right)\tilde{c}_k$$

with $q_k = 1$ if $k = 0$, $q_k = 0$ otherwise and $(\tilde{c}_0, \ldots \tilde{c}_d) = \mathrm{IDCT}((f_0, \ldots, f_d))$. Let $T_\alpha = T_{\alpha_1}(x_1) \cdots T_{\alpha_n}(x_n) \in \mathbb{C}[x_1, \ldots, x_n], \alpha \in \mathbb{N}^n$. 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, this generalizes as follows. We define an n-dimensional array $(f_k)_{k_1=0,\ldots,k_n=0}^{d_1,\ldots,d_n}$ of function values given by

$$f_k = f_{k_1,\ldots,k_n} = f(\omega_{k,d}) = f(\omega_{k_1,d_1}, \ldots, \omega_{k_n,d_n}).$$

We obtain another such array by performing an n-dimensional IDCT in the usual way: a series of 1-dimensional IDCTs along every dimension of the array. This gives $(\tilde{c}_\alpha)_{\alpha_1=0,\ldots,\alpha_n=0}^{d_1,\ldots,d_n}$ and the coefficients in the product Chebyshev basis are given by

$$c_\alpha = \left(\frac{1}{\sqrt{2}}\right)^{q_\alpha}\left(\prod_{i=1}^{n}\sqrt{\frac{2}{d_i+1}}\right)\tilde{c}_\alpha$$

with q_α the number of zero entries in α. This shows that the coefficients c_α needed to construct the matrix of Res can be computed efficiently by taking an IDCT of an array of function values of the monomial multiples of the f_i. The development of this technique is future research.

A situation in which it is natural to use a product Chebyshev basis $\mathcal{V}$ for V is when $f_i = 0$ are (local) approximations of real transcendental (or higher degree algebraic) hypersurfaces.

[4]We use the definitions of the discrete cosine transform that agree with the built in `dct` command in Matlab.

Chebyshev polynomials have remarkable interpolation and approximation properties on compact intervals of the real line, see Trefethen (2013). The multivariate product bases $\{T_\alpha\}$ inherit these properties for bounded boxes in $\mathbb{R}^n$. In Nakatsukasa et al. (2015), bivariate, real intersection problems are solved by local Chebyshev approximation, and this is what is implemented in the `roots` command of Chebfun2 (Townsend and Trefethen (2013)). If the ideal I is expected to have many real solutions in a compact box of $\mathbb{R}^n$, it is probably a good idea to represent the generators in the Chebyshev basis. One reason is that functions with a lot of real zeroes have 'nice coefficients' in this basis, whereas in the monomial basis, they don't. We work out an example of this in Section 6.

We conclude this subsection 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 (Szegő (1967)).

6. Numerical experiments

In this section we show some experimental results. The aim is twofold:

1. to show the potential of the TNF approach as an alternative for some state of the art polynomial system solvers, summarizing and extending the experiments in Telen et al. (2018),
2. to illustrate the techniques presented in this paper.

We use a Matlab implementation of the algorithms in Telen et al. (2018) and of the algorithms presented here to compute the multiplication tables. In most experiments, we then compute the roots from those tables[5]. For a description of how this second step works, see Corless et al. (1997); Möller and Tenberg (2001); Elkadi and Mourrain (2007). In a first subsection, we show how affine dense, affine sparse and homogeneous systems can be solved accurately using TNFs. In Subsection 6.2 we summarize the comparison in Telen et al. (2018) with the homotopy continuation packages PHCpack (Verschelde (1999)) and Bertini (Bates et al. (2013)). In Subsection 6.3 we compare the TNF algorithm to construct the multiplication matrices with a Groebner basis normal form method. We use Faugère's FGb (Faugère (2010)) 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. In Subsections 6.4, 6.5, 6.6 and 6.7 we illustrate the results from Sections 3, 4, Subsection 5.1 and 5.2 respectively. In all of the experiments, the *residual* is a measure for the backward error computed as in Telen and Van Barel (2018). Using double precision arithmetic, the best residual one can hope for is of order 10^{-16}. The experiments are performed on an 8 GB RAM machine with an intel Core i7-6820HQ CPU working at 2.70 GHz, unless stated otherwise.

6.1. Some nontrivial examples

6.1.1. Intersecting two plane curves of degree 170

Consider all monomials of $\mathbb{C}[x_1, x_2]$ of degree $\leq d$ and assign a (floating point, double precision) coefficient to each of these monomials drawn from a normal distribution with mean 0 and

[5]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.

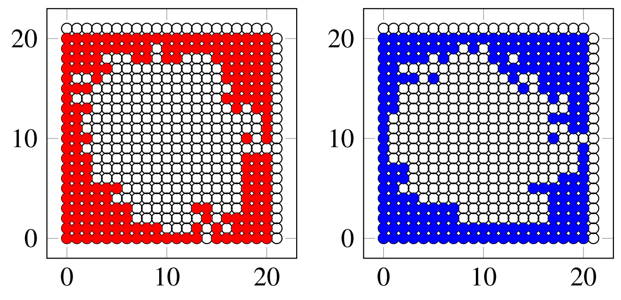


Figure 2: Monomials spanning V (○) and monomials in the basis for system 1 (●) and system 2 (●).

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 $\mathbb{C}^2$. The curves intersect in $\delta = d^2$ points, according to Bézout's theorem. To show the potential of the TNF approach, 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. We note that a polynomial of degree $d = 170$ in two variables has 14706 terms.

6.1.2. 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 2 from Telen et al. (2018) 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 (Joswig et al. (2009)). We used QR with optimal column pivoting on $N_{|W}$ for the basis choice. Figure 2 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 $\mathbb{N}^2$. 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.

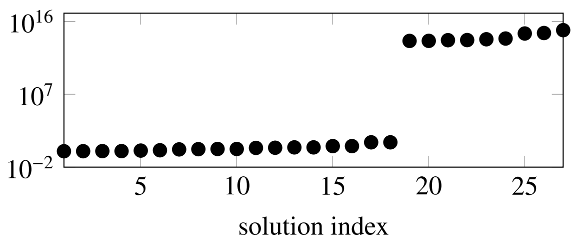


Figure 3: Norms of the computed solutions of 3 homogeneous equations in 4 variables in the affine chart $x_0 = 1$.

6.1.3. Solutions at infinity

As shown in Telen et al. (2018) (Algorithm 3), TNFs can be used to solve homogeneous systems defining points in $\mathbb{P}^n$. Consider 3 dense homogeneous equations f_1, f_2, f_3 in $\mathbb{C}[x_0, \ldots, x_3]$ 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 3 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.

6.2. Comparison with homotopy solvers

The homotopy continuation packages PHCpack and Bertini are standard tools for solving a system of polynomial equations (Verschelde (1999); Bates et al. (2013)). 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 Telen et al. (2018); Telen and Van Barel (2018) we learn that an advantage of algebraic methods over homotopy continuation methods is that they guarantee to find numerical approximations of *all* solutions. The homotopy packages (using standard double precision settings) tend to give up on some of the paths once the systems become of larger degree and consistently miss some solutions. Table 2 illustrates this for $n = 2$ variables and degrees $d \geq 25$ (see tables in Subsection 8.5 of Telen et al. (2018) 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 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 TNF approach (and in general, of all algebraic approaches) is that its complexity scales badly with the number of variables n, as explained in section 4. Although the TNF solver 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. We show in Subsection 6.5 that the techniques introduced in Section 4 can be used to push these cross-overs back to higher degrees.

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.

6.3. Comparison with Groebner bases

In this subsection we compare the TNF 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*', see Cox et al. (1992). 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 the TNF algorithm.

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

Table 3: Corresponding steps of the TNF algorithm and the Groebner basis algorithm

We have used Faugère's FGb in Maple for step 1 (Faugère (2010)). This is considered state of the art software for computing 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 $\mathscr{B}$, as shown before, and they are computed in finite precision. Of course, a speed-up with respect to exact arithmetic is to be expected. The goal of this experiment is to quantify this speed-up and the price we pay for this speed-up (i.e. a numerical approximation error on the computed result). 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 residuals are never larger than order 10^{-10} 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 compute the eigenvalues of these multiplication operators by using a numerical method. This solving step is not integrated in the comparison.

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.

6.3.1. 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 TNFs 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 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.

6.3.2. 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	$5.68 \cdot 10^{-4}$	$1.52 \cdot 10^{-2}$	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

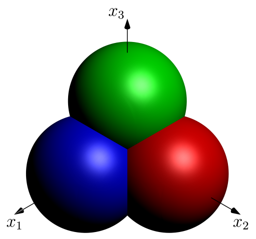
Table 4: Timing results for the TNF algorithm (t_{TNF} (sec)) and the Groebner basis algorithm in Maple (t_{GB} (sec)) for generic systems in n variables of degree d with floating point coefficients drawn from a normal distribution with zero mean and $\sigma = 1$.

6.4. Solving non-generic systems

6.4.1. Intersecting three spheres

To illustrate the algorithm proposed in Section 3, we consider the following example. Let $I = \langle f_1, f_2, f_3 \rangle \subset R = \mathbb{C}[x_1, x_2, x_3]$ be given by

$$\begin{aligned} f_1 &= x_1^2 - 2x_1 + x_2^2 + x_3^2, \\ f_2 &= x_1^2 + x_2^2 - 2x_2 + x_3^2, \\ f_3 &= x_1^2 + x_2^2 + x_3^2 - 2x_3. \end{aligned}$$



The equation $f_i = 0$ represents a sphere of radius 1 centered at $e_i \in \mathbb{C}^3$. The Bézout number of three quadratic surfaces is 8. However, there are only 2 solutions: $\mathbb{V}(I) = \{(0,0,0), (\frac{2}{3}, \frac{2}{3}, \frac{2}{3})\}$. Both solutions are regular. Homogenizing the equations we note that the f_i define a curve at infinity. This means that using the projective version of the TNF algorithm in Telen et al. (2018) will not solve this problem: it assumes finitely many solutions in $\mathbb{P}^3$. We use Algorithm 1 to find the roots. To this end, note that $\{1 + I, x_i + I\}$ is a basis for R/I. The vector spaces $W' = R_{\leq 1}, V' = R_{\leq 2}$ satisfy the conditions 1-3 of Theorem 5. We consider the resultant map $\mathrm{Res} : (R_{\leq 1})^3 \to R_{\leq 3} : (q_1, q_2, q_3) \mapsto q_1 f_1 + q_2 f_2 + q_3 f_3$. The cokernel N of Res has rank 9: $N : R_{\leq 3} \to \mathbb{C}^9$, yet $N_{|W}$ has rank $r_{V'} = 2$ ($W = V' = R_{\leq 2}$). We used the monomial basis for all the computations. A column pivoted QR outputs $\mathscr{B}' = \{1, x_3\}$ as (representatives of) a monomial basis for R/I. Algorithm 1 finds the two solutions with a *forward error* of order 10^{-16} as the eigenvalues of a generalized pencil of four 2×2 matrices. Note that the standard Macaulay construction (presented in Section 4) gives $V = R_{\leq 4}$, which shows that Algorithm 1

n	d	t_{TNF}	t_{GB}	$t_{\text{GB}}/t_{\text{TNF}}$
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 TNF algorithm (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.

may lead to smaller matrices than the standard constructions in the case of systems with $\delta' < \delta$ solutions. Note that in this example, it is not sufficient to take $V = V' = R_{\leq 2}$ and $W' = R_{\leq 1}$ for the resultant construction, because this gives a resultant map with cokernel onto $\mathbb{C}^7$ and $\dim_{\mathbb{C}}(W') = 4 < \operatorname{rank}(N_{|W'}) = 7$.

6.4.2. Intersecting two quartics with a common factor

We illustrate how Algorithm 1 can find the isolated points of a variety containing a one-dimensional irreducible component. To this end, we consider the ideal $I = \langle f_1, f_2 \rangle \subset R = \mathbb{C}[x_1, x_2]$ defined by

$$\begin{aligned} f_1 &= x_1^2x_2 + x_2^3 - x_2 - x_1^4 - x_1^2x_2^2 + x_1^2 = (x_1^2 + x_2^2 - 1)(x_2 - x_1^2), \\ f_2 &= x_1^2x_2 + x_2^3 - x_2 + x_1^4 + x_1^2x_2^2 - 9x_1^2 - 8x_2^2 + 8 = (x_1^2 + x_2^2 - 1)(x_2 + x_1^2 - 8). \end{aligned}$$

It is clear that $\mathbb{V}(I)$ is the union of the unit circle and the two intersection points $\{(\pm 2, 4)\}$ of two parabolas. We use Algorithm 1 with $\rho = 3$, $V = R_{\leq \rho+2}$, $V' = W = R_{\leq \rho+1}$, $W' = R_{\leq \rho}$. This leads to a regular pencil of size 9. Only two of the eigenvalues correspond to solutions, and the computed solutions are $\{(\pm 2, 4)\}$ up to a forward error of order 10^{-16}.

6.5. *Efficient construction of TNFs*

Define a generic system of degree d in n variables as in 6.3.1. In this subsection, we illustrate the techniques presented in Section 4 to speed up the TNF computation. Table 6 gives the results. In the table we present the computation times t and the maximal residuals r of three different algorithms: TNF stands for the standard TNF algorithm, FM stands for the algorithm suggested in Subsection 4.2 using fewer multiples of the input equations for the construction of Res and DBD represents the algorithm from Subsection 4.1 which computes the cokernel degree by degree. For all of the algorithms, we used pivoted QR for the basis selection. For $n = 2$, both

n	d	t_{TNF} (sec)	$t_{\mathrm{TNF}}/t_{\mathrm{FM}}$	$t_{\mathrm{TNF}}/t_{\mathrm{DBD}}$	re_{TNF}	re_{FM}	re_{DBD}
3	2	$1.57 \cdot 10^{-2}$	1.46	0.21	$8.95 \cdot 10^{-16}$	$2.19 \cdot 10^{-15}$	$8.44 \cdot 10^{-16}$
3	3	$4.67 \cdot 10^{-2}$	1.24	0.89	$3.02 \cdot 10^{-15}$	$4.65 \cdot 10^{-14}$	$1.55 \cdot 10^{-15}$
3	4	0.1	1.04	1.35	$1.19 \cdot 10^{-14}$	$2.76 \cdot 10^{-14}$	$8.76 \cdot 10^{-15}$
3	5	0.17	1.06	0.96	$1.43 \cdot 10^{-14}$	$5.14 \cdot 10^{-13}$	$4.92 \cdot 10^{-15}$
3	6	0.41	1.03	0.95	$5.16 \cdot 10^{-15}$	$9.48 \cdot 10^{-14}$	$7.06 \cdot 10^{-15}$
3	7	1.67	1.19	1.47	$8.82 \cdot 10^{-15}$	$1 \cdot 10^{-13}$	$4.05 \cdot 10^{-14}$
3	8	6.23	1.16	2.04	$1.19 \cdot 10^{-13}$	$6.71 \cdot 10^{-11}$	$5.64 \cdot 10^{-14}$
3	9	18.03	1.16	2.61	$2.3 \cdot 10^{-13}$	$6.58 \cdot 10^{-12}$	$2.54 \cdot 10^{-14}$
3	10	45.81	1.16	2.99	$1.56 \cdot 10^{-13}$	$5.67 \cdot 10^{-12}$	$7.08 \cdot 10^{-14}$
3	11	56.36	1.06	1.57	$1.16 \cdot 10^{-13}$	$1.81 \cdot 10^{-12}$	$2.14 \cdot 10^{-13}$
3	12	117.31	1.17	1.55	$1.83 \cdot 10^{-13}$	$3.21 \cdot 10^{-12}$	$8.35 \cdot 10^{-14}$
3	13	229.96	1.16	1.58	$3.16 \cdot 10^{-13}$	$8.87 \cdot 10^{-11}$	$2.03 \cdot 10^{-12}$
4	2	$3.81 \cdot 10^{-2}$	1.39	1.24	$1.36 \cdot 10^{-14}$	$2.35 \cdot 10^{-12}$	$2.74 \cdot 10^{-15}$
4	3	0.28	1.06	1.23	$1.55 \cdot 10^{-13}$	$2.91 \cdot 10^{-13}$	$1.67 \cdot 10^{-14}$
4	4	10.05	1.46	4.42	$5.82 \cdot 10^{-15}$	$1.36 \cdot 10^{-12}$	$1 \cdot 10^{-14}$
4	5	147.32	2.61	5.77	$9.97 \cdot 10^{-14}$	$6.6 \cdot 10^{-13}$	$5.47 \cdot 10^{-14}$
5	2	0.15	1.04	1.12	$3.58 \cdot 10^{-15}$	$9.38 \cdot 10^{-14}$	$1.8 \cdot 10^{-15}$
5	3	75.37	2.78	4.64	$1.97 \cdot 10^{-14}$	$1.83 \cdot 10^{-12}$	$3.49 \cdot 10^{-14}$
6	2	3.44	1.24	1.7	$1.91 \cdot 10^{-15}$	$2.46 \cdot 10^{-13}$	$3.66 \cdot 10^{-15}$
7	2	167.53	1.96	2.41	$1.69 \cdot 10^{-14}$	$4.01 \cdot 10^{-11}$	$3.07 \cdot 10^{-14}$

Table 6: Timing and relative error for the variants of the TNF algorithm presented in Section 4 for generic systems in n variables of degree d.

alternatives don't give any improvements. As shown earlier, the TNF algorithm is very efficient as it is in this case. For $n > 2$ we see that both FM and DBD can make the algorithm significantly faster for sufficiently high degrees, and not much (or none) of the accuracy is lost. The biggest speed-up we achieved in the experiment is a factor 5.77 for $n = 4, d = 5$. Solving such a system takes about 17 seconds using Bertini and 11 seconds using PHCpack. PHCpack loses 2 out of 625 solutions. The DBD algorithm takes less than 26 seconds to find all solutions with a residual no larger than $\pm 10^{-14}$. The unmodified TNF algorithm takes 3 to 4 times as much time as the homotopy solvers for $n = 4, d = 4$ (see the experiments in Telen et al. (2018)). The DBD algorithm is as fast as PHCpack, which is 1.6 times faster than Bertini in this case. The algorithms do not beat the homotopy solvers for larger numbers of variables, even in small degrees. For $n = 7, d = 2$, both homotopy packages solve the problem in less than 4 seconds, while the fastest version of the TNF solver takes more than a minute.

To compare the FM algorithm with the classical Macaulay resultant construction where the V_i are replaced by the span of a specific subset of monomials (see (Cox et al., 2006, Chapter 3)), we used this construction to solve the case $n = 3, d = 13$. The obtained residual was $1.44 \cdot 10^{-4}$, which is roughly a factor 10^7 larger than the re_{FM}.

6.6. *Using SVD for the basis selection*

We use the toric variant of the TNF algorithm to compute the 24 real solutions of a complete intersection in $R = \mathbb{C}[x_1, x_2]$. For the basis selection, we use the singular value decomposition instead of QR as explained in Subsection 5.1. The real curves defined by the generators of I and the solutions are depicted in Figure 4. A qualitative picture of the resulting 24 basis

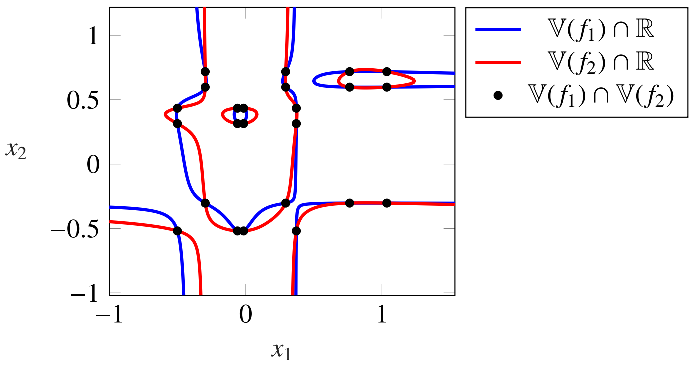


Figure 4: Real algebraic curves defined by f_1, f_2 and solutions of $I = \langle f_1, f_2 \rangle$ from Subsection 6.6.

elements in $\mathcal{B} = \{b_1, \ldots, b_\delta\}$ is shown in Figure 5. We show some contour lines on the real plane. Dark (blue) colours represent small absolute values of b_i, yellow colours correspond to high values. As monomials only vanish on the axes, we see that the obtained basis functions behave fundamentally differently. Especially the last basis functions (lower part of the figure) show some interesting action near the roots. We leave the possible relation between the root location and the orthogonal basis functions for future research. The residual using SVD in this example is $5.16 \cdot 10^{-12}$, for QR it is $2.84 \cdot 10^{-11}$.

6.7. *TNFs in the product Chebyshev basis*

In this experiment we illustrate the use of Chebyshev polynomials in the construction of a TNF. To this end, we construct a polynomial system as follows. We fix a degree d and define $f_1 = \sum_{|\alpha| \le d} c_{\alpha,1} T_\alpha, f_2 = \sum_{|\alpha| \le d} c_{\alpha,2} T_\alpha$ where $T_\alpha = T_{\alpha_1}(x_1) T_{\alpha_2}(x_2)$ as in Subsection 5.2 and the $c_{\alpha,i}$ are drawn from a zero mean, $\sigma = 1$ normal distribution. Since the zeroes of T_i are all in the real interval $[-1, 1]$, we expect interesting things to happen in the box $[-1, 1] \times [-1, 1] \subset \mathbb{R}^2$ for the curves defined by f_1, f_2, so we expect a large number of real roots in a bounded region of $\mathbb{R}^2$. This is the situation in which we expect the Chebyshev basis to have good numerical properties. For $d = 20$, we computed the solutions using a TNF with QR for basis selection in the monomial basis and in the Chebyshev basis. The residuals of all 400 solutions are represented in Figure 6 in the form of a histogram. As expected, the Chebyshev TNF performs better. The TNF in the monomial basis still gives acceptable results: the largest residual is of order 10^{-6}. If we increase the degree to $d = 25$, the difference in performance grows. There are 625 solutions in this case. Results are shown in Figure 7 and the curves are depicted in Figure 8. Using monomials, one solution has residual of order 10^{-1}, which means we basically lost this solution.

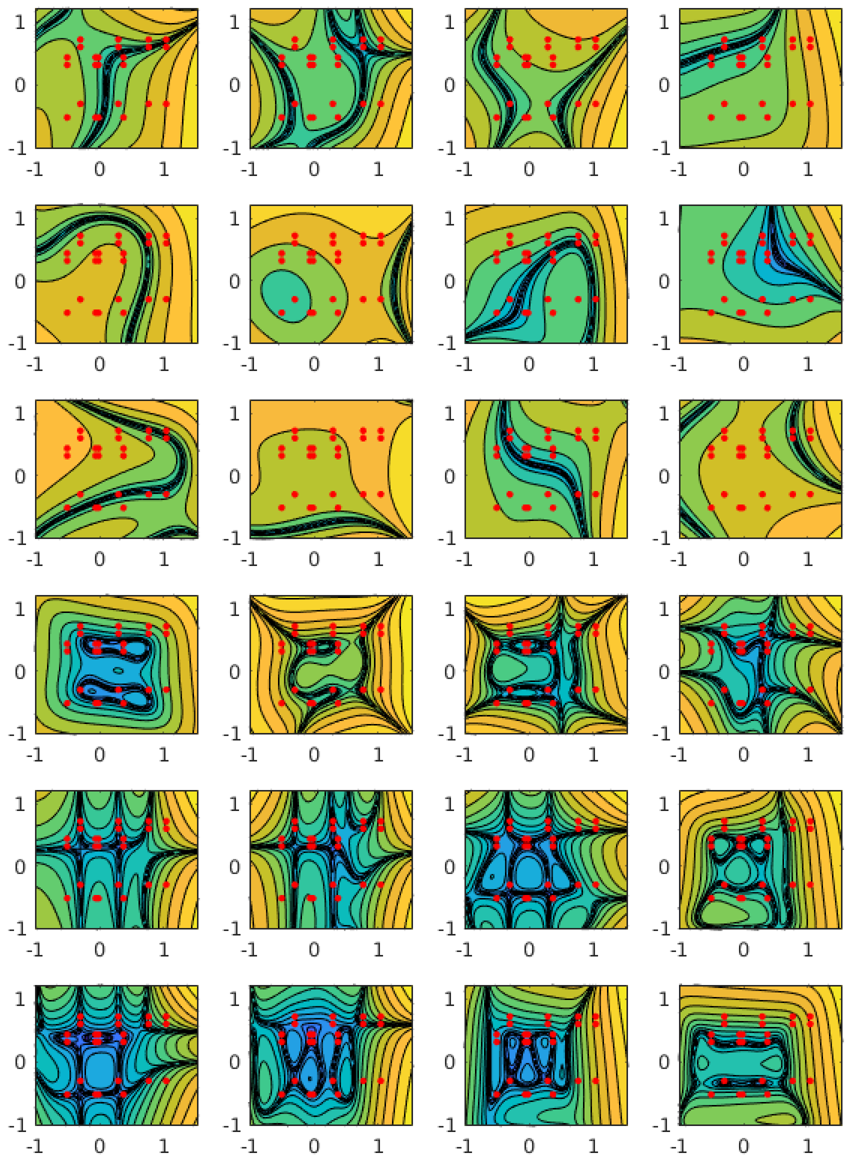


Figure 5: Orthogonal basis for R/I computed using the SVD on $N_{|W}$. The red dots are the roots of I.

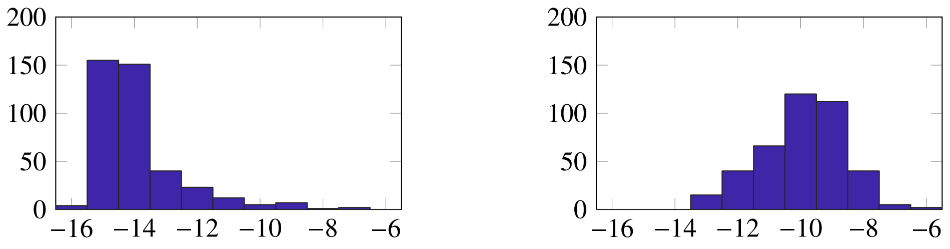


Figure 6: Histogram of $\log_{10}$ of the backward error for a system as described in Subsection 6.7 of degree 20 using the Chebyshev basis (left) and the monomial basis (right).

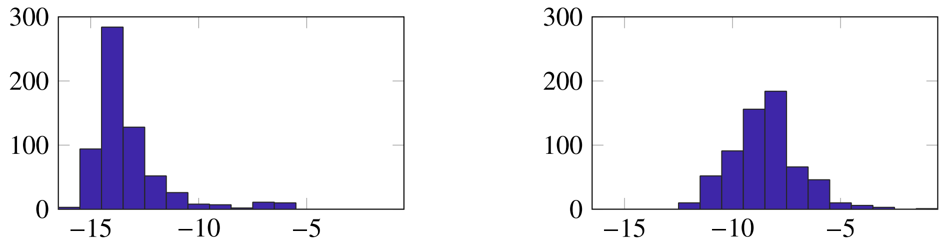


Figure 7: Histogram of $\log_{10}$ of the backward error for a system as described in Subsection 6.7 of degree 25 using the Chebyshev basis (left) and the monomial basis (right).

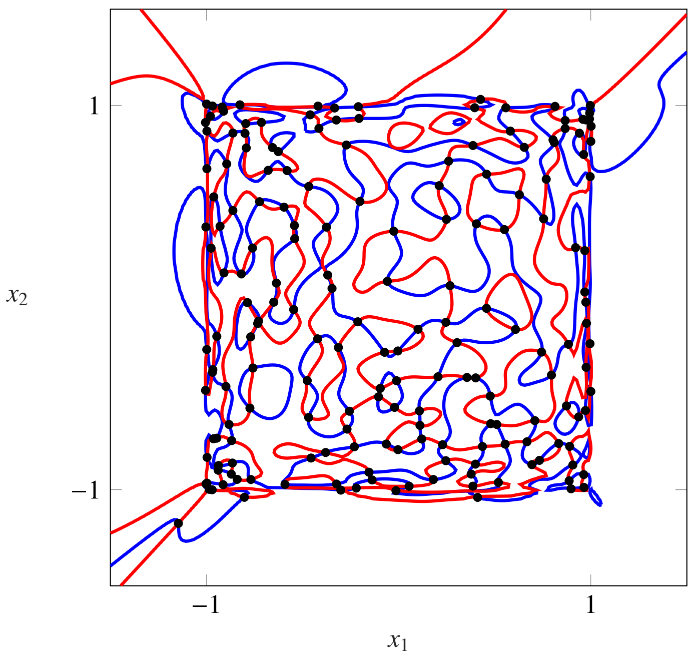


Figure 8: Real picture of a degree 25 system as described in Subsection 6.7.

7. Conclusion

We have presented generalized and more efficient versions of the truncated normal form algorithm for solving systems of polynomial equations. More precisely, we presented an algorithm for solving non-generic systems based on TNFs, proposed fast algorithms for computing cokernel maps of resultant maps and we illustrated the flexibility to choose the type of basis functions for the quotient algebra in function of, for instance, where the roots are expected to be. The experiments show that the TNF method is competitive with the state of the art algebraic and homotopy based solvers and that the contributions of this paper can lead to significant improvements of the accuracy and efficiency.

References

Bates, D. J., Hauenstein, J. D., Sommese, A. J., Wampler, C. W., 2013. Numerically solving polynomial systems with Bertini. Vol. 25. SIAM.

Batselier, K., Dreesen, P., De Moor, B., 2014. A fast recursive orthogonalization scheme for the macaulay matrix. Journal of Computational and Applied Mathematics 267, 20–32.

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

Corless, R. M., Gianni, P. M., Trager, B. M., 1997. 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. ACM, pp. 133–140.

Cox, D. A., Little, J., O'Shea, D., 1992. Ideals, varieties, and algorithms. Vol. 3. Springer.

Cox, D. A., Little, J., O'Shea, D., 2006. Using algebraic geometry. Vol. 185. Springer Science & Business Media.

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

Elkadi, M., Mourrain, B., 2007. Introduction à la résolution des systèmes polynomiaux. Vol. 59 of Mathématiques et Applications. Springer.

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

Faugère, J.-C., September 2010. FGb: A Library for Computing Groebner Bases. In: Fukuda, K., Hoeven, J., Joswig, M., Takayama, N. (Eds.), Mathematical Software - ICMS 2010. Vol. 6327 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Berlin, Heidelberg, pp. 84–87.

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

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

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

Mourrain, B., 2007. Pythagore's dilemma, symbolic-numeric computation, and the border basis method. In: Symbolic-Numeric Computation. Springer, pp. 223–243.

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

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

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

Nakatsukasa, Y., Noferini, V., Townsend, A., 2015. Computing the common zeros of two bivariate functions via bézout resultants. Numerische Mathematik 129 (1), 181–209.

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

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

Sturmfels, B., 2002. Solving Systems of Polynomial Equations. No. 97 in CBMS Regional Conferences. Amer. Math. Soc.

Szegő, G., 1967. Orthogonal Polynomials: 3d Ed. American Mathematical Society.

Telen, S., Mourrain, B., Barel, M. V., 2018. Solving polynomial systems via truncated normal forms. SIAM Journal on Matrix Analysis and Applications 39 (3), 1421–1447.

Telen, S., Van Barel, M., 2018. A stabilized normal form algorithm for generic systems of polynomial equations. Journal of Computational and Applied Mathematics 342, 119–132.

Townsend, A., Trefethen, L. N., 2013. An extension of chebfun to two dimensions. SIAM J. Scientific Computing 35.

Trefethen, L. N., 2013. Approximation theory and approximation practice. Vol. 128. Siam.

Trefethen, L. N., Bau III, D., 1997. Numerical linear algebra. Vol. 50. Siam.

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