Eigensolvers for polynomial roots and tensor decomposition

Enrica Barrilli and Bernard Mourrain

Centre Inria at Université Côte d'Azur

Abstract. Computing eigenvalues and eigenvectors is at the heart of the solution of many non-linear problems. For instance, finding the roots of polynomial systems reduces to computing joint eigenvectors of operators of multiplication. Similarly, tensor decomposition can be performed via the joint diagonalization of submatrices of the Catalecticant of the tensor. We describe and illustrate symbolic-numeric methods for computing the solutions of these algebraic problems from the computation of joint eigenvectors of commuting operators, and for analysing their multiplicity structure, as well as their implementation in the package AlgebraicSolvers.jl

1 Introduction

Nonlinear problems are ubiquitous in scientific computing. However, solving them efficiently often relies on the use of fundamental tools from linear algebra, such as solving linear systems, or computing eigenvalues or eigenvectors.

In this article, we aim to illustrate this "motto", focusing on the solutions of algebraic problems. A typical algebraic problem is finding all the roots of a set of polynomial equations. It has applications in many domains, e.g. in robotics, computer vision, chemistry, signal processing, polynomial optimization, ... (see e.g. [34,15,22,9,23,13]).

Various methods exist to find the roots of polynomial equations. The most popular are probably local iterative methods, such as Newton-Raphson method [11], which start from an initial point and usually converge to a fixed point. Such methods are very efficient in practice to find a single solution, but the convergence is not guaranteed and depends highly on the initialization. Moreover, they find one solution and have difficulties to find all the solutions of a system.

Homotopy solvers constitute another family of solvers [34]. They exploit local iterative methods in a controlled way, by deforming a system with known solutions into the system to be solved, tracking the paths of the solutions by local methods. They are efficient, but can suffer from the proximity of singularities.

We are interested in Algebraic solvers, which exploit the properties of the quotient algebra $\mathcal{A} = R/I$ of the ring of polynomials R by the ideal I generated by the system of equations to solve [10]. They make it possible to calculate all the roots of a zero-dimensional system and to analyze their multiplicities.

The AlgebraicSolvers.jl package is dedicated to this family of methods. It is based on the concept of Truncated Normal Form (TNF) [29,36,32,31], which provides a uniform approach to handling this family of solvers. Tools for computing TNFs based on the computation of Gröbner bases are available (see Section 3.1). This computation has been optimised over several decades [17,14,4,12] to achieve a high level of performance, but it applies to polynomial equations with exact coefficients (for example, coefficients in $\mathbb{Q}$). However the TNF can be derived using numerical linear algebra (with floating-point arithmetic), once the Gröbner basis has been computed for any order. Border bases, developed to address the problem of numerical instability in Gröbner bases [29,33,21], can be used in the same way to compute a TNF.

Resultant matrix constructions, which have also been devised to solve specific classes of polynomial systems (e.g. generic polynomials with a given support) [16,9] can also be used to deduce a TNF. The TNF is computed from the co-kernel of the resultant matrix, again using numerical linear algebra. In the package AlgebraicSolvers.jl, the solvers are parametrized by classes, which specify how to compute the TNF (see Section 3).

TNFs can also be used in Tensor Decomposition Problems (see [20,5,3] and Section 3.2) or in Polynomial Optimization Problems [23] for recovering the optimizers. They can be computed

directly from the Hankel or Catalecticant matrix associated to the tensor or from the optimal moment matrix in Polynomial Optimization Problems (see [19,31]).
Truncated Normal Forms provide an effective description of the multiplicative structure of the quotient algebra $\mathcal{A}$ and, in particular, of the operators of multiplication by elements in $\mathcal{A}$. The solutions of the polynomial systems or the nodes in a tensor decomposition problem can then be recovered by eigencomputation [1,27,28,35,15]. We recall these properties in Section 2 and illustrate the TNF approach in Section 3. For simple roots, their coordinates are computed by join diagonalization of the matrices of multiplication by the variables. Direct eigenvector computation (using the function `LinearAlgebra.eigen`) of a random combination of the matrices or iterative numerical methods (see [37,7,6]) are implemented. In the presence of multiple roots, a Schur factorization is performed (using `LinearAlgebra.schur`) and the coordinates of the roots are recovered using traces of the diagonal blocks of the multiplication matrices (see [8,2]).
To analyze further the multiplicity of a root, inverse systems can also be computed, using the function `AlgebraicSolvers.invsys` based on linear algebra tools (see [24,28,18,26,25]).
These methods are based on the use of duality properties on polynomial rings and on the representation of linear functionals as formal power series (see `AlgebraicSolvers.Series`).

2 The structure of an Artinian algebra

An *Artinian* algebra $\mathcal{A}$ is a commutative $\mathbb{K}$-algebra of finite dimension as a $\mathbb{K}$ vector space: $\dim_{\mathbb{K}} \mathcal{A} < \infty$. As it is finite dimensional, $\mathcal{A}$ is also a finitely generated algebra. Suppose that the Artinian algebra $\mathcal{A}$ is generated by $a_1, \ldots, a_n$ as an algebra. Then we have the exact sequence:

$$\begin{array}{ccccccc} 0 & \longrightarrow & I & \longrightarrow & \mathbb{K}[x_1, \ldots, x_n] & \longrightarrow & \mathcal{A} \longrightarrow 0 \\ & & & & x_i & \longmapsto & a_i \end{array}$$

where I is a zero-dimensional ideal in the ring R of polynomials in n variables $\mathbb{K}[\mathbf{x}] = \mathbb{K}[x_1, \ldots, x_n]$ such that $\mathcal{A} = \mathbb{K}[\mathbf{x}]/I$ is of finite dimension. Conversely, any zero-dimensional ideal $I \subset \mathbb{K}[\mathbf{x}]$ defines the Artinian algebra $\mathcal{A} = R/I$.
The algebraic set $\mathcal{V}_{\overline{\mathbb{K}}}(I) = \{\xi \in \overline{\mathbb{K}}^n \mid \forall p \in I, p(\xi) = 0\}$ is finite iff I is zero dimensional, iff $\mathbb{K}[\mathbf{x}]/I$ is Artinian, where $\overline{\mathbb{K}}$ is the algebraic closure of $\mathbb{K}$ (see [10]).
Let $r = \dim_{\mathbb{K}} \mathcal{A}$, $\{\xi_1, \ldots, \xi_{r'}\} = \mathcal{V}_{\overline{\mathbb{K}}}(I)$ with $r' \leq r$. Let us assume that $\mathbb{K} = \overline{\mathbb{K}}$ is algebraically closed and let $I = \cap_{i=1}^{r'} Q_i$ be a primary decomposition of the ideal I, where Q_i is a primary ideal for the maximal ideal $\mathbf{m}_{\xi_i}$ defining ξ_i. Then $\mathcal{A}$ decomposes as

$$\mathcal{A} = \mathcal{A}_1 \oplus \cdots \oplus \mathcal{A}_{r'}$$

where $\mathcal{A}_i = \mathbf{u}_i \mathcal{A} \sim R/Q_i$ is of dimension $\mu_i \leq r$ and $\mathbf{u}_1, \ldots \mathbf{u}_{r'}$ is the family of orthogonal idempotents of maximal size, satisfying $\mathbf{u}_i^2 = \mathbf{u}_i$ for $i = 1, \ldots, r'$, $\quad \sum_{i=1}^{r'} \mathbf{u}_i = 1$. The dimension $\dim_{\mathbb{K}} \mathcal{A}_i$ is called the *multiplicity* of ξ_i (see [15]).
Given $p \in R$ (or $p \in \mathcal{A}$ identifying $p \in R$ with its class in $\mathcal{A} = R/I$), we define the operator M_p of multiplication by p and its transposed M_p^* as

$$\begin{array}{ccc} M_p : \mathcal{A} \longrightarrow \mathcal{A} & \qquad & M_p^* : \mathcal{A}^* \longrightarrow \mathcal{A}^* \\ a \longmapsto p\, a & & \Lambda \longmapsto p \star \Lambda \end{array}$$

where $\mathcal{A}^* = \mathrm{Hom}_{\mathbb{K}}(\mathcal{A}, \mathbb{K})$ is the dual of $\mathcal{A}$ and $p \star \Lambda : q \in \mathcal{A} \longmapsto \Lambda(p\, q) \in \mathbb{K}$.
We use the following classical theorem (see [1,27,28,15,35]) for solving polynomial systems:

Theorem 2.1. *For any $p \in R = \mathbb{K}[\mathbf{x}]$,*

- *the eigenvalues of M_p repeated with their multiplicities, are $\{\overbrace{p(\xi_1), \ldots, p(\xi_1)}^{\mu_1}, \ldots, \overbrace{p(\xi_{r'}), \ldots, p(\xi_{r'})}^{\mu_{r'}}\}$, where μ_i is the multiplicity of the root ξ_i.*
- *The common eigenvectors of all $(M_p^*)_{p \in R}$ are, up to a scalar, the evaluation functionals $\mathbf{e}_{\xi_i} : q \in R \mapsto q(\xi_i)$.*

This allows us to recover the roots $\Xi = \{\xi_1, \ldots, \xi_{r'}\}$ by eigencomputation, using the operators of multiplication M_{x_i} for $i = 1, \ldots, n$. In particular, when the roots are simple, we have the following (see e.g. [15]):

Proposition 2.2. *If the roots* $\{\xi_1, \ldots, \xi_r\} = \mathcal{V}_{\overline{\mathbb{K}}}(I)$ *are* simple *(i.e.* $\mu_i = 1$*),*

- *The operators* M_{x_i} *for* $i = 1, \ldots, n$ *have a joint diagonalization.*
- *Their common eigenvectors are, up to a scalar, the* interpolation polynomials $\mathbf{u}_i$ *at the roots, satisfying* $\mathbf{u}_i(\xi_j) = \delta_{i,j}$*. They form a basis of orthogonal idempotents of* $\mathcal{A}$ *(i.e.* $\mathbf{u}_i^2 \equiv \mathbf{u}_i$*,* $\mathbf{u}_i\mathbf{u}_j \equiv 0$ *if* $i \neq j$*,* $\sum_{i=1}^{r} \mathbf{u}_i \equiv 1$*).*
- *The common eigenvectors of the transpose of the matrices of* $(M_{x_i})_{i=1,\ldots,n}$ *in a basis* B *of* $\mathcal{A}$ *are, up to a scalar, the vectors* $B(\xi_j)$ *representing the evaluation linear functionals* $\mathbf{e}_{\xi_1}, \ldots, \mathbf{e}_{\xi_r}$ *in the basis of* $\mathcal{A}^* = I^\perp$ *dual to* B*.*

When the roots are multiple, we have the following theorem:

Proposition 2.3. *There exists a basis of* $\mathcal{A}$ *in which, all the operators of multiplication* M_p *for* $p \in R$ *decompose as* $\mathrm{diag}(M_p^1, \ldots, M_p^{r'})$ *where* M_p^i *is the matrix of the operator*

$$M_p^i : a \in \mathcal{A}_i \longmapsto p\,a \in \mathcal{A}_i$$

with $\mathrm{Trace}(M_p^i) = \mu_i\, p(\xi_i)$ *and* $\mu_i = \dim(\mathcal{A}_i)$ *for* $i = 1, \ldots, r'$*.*

Such a basis is a basis adapted to the decomposition $\mathcal{A} = \oplus_{i=1}^{r'} \mathcal{A}_i$, i.e. of the form $\mathbf{u}_i\, b_{i,j}$ where $(b_{i,j})_j$ is a basis of $\mathcal{A}_i$ and $\mathbf{u}_i$ is the idempotent defining $\mathcal{A}_i$. See [8,15] for more details.

In order to analyse the multiplicity structure of the roots, we introduce the following notation. Let $\mathbb{K}[[y_1, \ldots, y_n]] = \mathbb{K}[[\mathbf{y}]]$ be the ring of formal power series in the variables $y_1, \ldots, y_n$ with coefficients in the field $\mathbb{K}$. To simplify notation, we assume hereafter that $\mathbb{K}$ *is of characteristic* 0.

There is a natural isomorphism between the ring of formal power series and the dual $R^* = \mathrm{Hom}_{\mathbb{K}}(R, \mathbb{K})$ of $R = \mathbb{K}[x_1, \ldots, x_n]$. It is given by the following pairing:

$$\begin{aligned} \mathbb{K}[[y_1, \ldots, y_n]] \times \mathbb{K}[x_1, \ldots, x_n] &\to \mathbb{K} \\ (\mathbf{y}^\alpha, \mathbf{x}^\beta) &\mapsto \langle \mathbf{y}^\alpha | \mathbf{x}^\beta \rangle = \begin{cases} \alpha! \text{ if } \alpha = \beta \\ 0 \quad \text{otherwise.} \end{cases} \end{aligned}$$

where $\alpha! = \prod_{i=1}^{n} \alpha_i!$ for $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. Namely, if $\Lambda \in R^* = \mathrm{Hom}_{\mathbb{K}}(R, \mathbb{K})$ is an element of the dual of $R = \mathbb{K}[\mathbf{x}]$, it can be represented by the series:

$$\Lambda(\mathbf{y}) = \sum_{\alpha \in \mathbb{N}^n} \Lambda(\mathbf{x}^\alpha) \frac{\mathbf{y}^\alpha}{\alpha!} \in \mathbb{K}[[\mathbf{y}]], \tag{1}$$

so that we have $\langle \Lambda(\mathbf{y}) \mid \mathbf{x}^\alpha \rangle = \Lambda(\mathbf{x}^\alpha)$. The map $\Lambda \in R^* \mapsto \sum_{\alpha \in \mathbb{N}^n} \Lambda(\mathbf{x}^\alpha) \frac{\mathbf{y}^\alpha}{\alpha!} \in \mathbb{K}[[\mathbf{y}]]$ is an isomorphism (see [30] for more details) and any series $\Lambda(\mathbf{y}) = \sum_{\alpha \in \mathbb{N}^n} \Lambda_\alpha \frac{\mathbf{y}^\alpha}{\alpha!} \in \mathbb{K}[[\mathbf{y}]]$ can be interpreted as a linear functional

$$p = \sum_{\alpha \in A \subset \mathbb{N}^n} p_\alpha \mathbf{x}^\alpha \in \mathbb{K}[\mathbf{x}] \mapsto \langle \Lambda \mid p \rangle = \sum_{\alpha \in A \subset \mathbb{N}^n} p_\alpha \Lambda_\alpha.$$

The coefficients $\Lambda_\alpha = \langle \Lambda \mid \mathbf{x}^\alpha \rangle$ for $\alpha \in \mathbb{N}^n$ are called the *pseudo-moments* of Λ. We identify the dual $R^* = \mathrm{Hom}_{\mathbb{K}}(\mathbb{K}[\mathbf{x}], \mathbb{K})$ with $\mathbb{K}[[\mathbf{y}]]$. Using this identification, the dual basis, denoted $(\mathbf{y}^{[\alpha]})_{\alpha \in \mathbb{N}^n}$, of the monomial basis $(\mathbf{x}^\alpha)_{\alpha \in \mathbb{N}^n}$ is $\left(\frac{\mathbf{y}^\alpha}{\alpha!}\right)_{\alpha \in \mathbb{N}^n} : \mathbf{y}^{[\alpha]} := \frac{\mathbf{y}^\alpha}{\alpha!}$.

From the definition, we verify that $\mathbf{y}^\alpha \in R^*$ acts on the polynomials by derivation: $\forall p \in R, \langle \mathbf{y}^\alpha \mid p \rangle = \partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha_n}(p)(0, \ldots, 0)$.

To illustrate the series representation, consider the evaluation at a point $\xi = (\xi_1, \ldots, \xi_n) \in \mathbb{K}^n$: $\mathbf{e}_\xi : p \in R \longmapsto p(\xi) \in \mathbb{K}$. It is represented by the series:

$$\mathbf{e}_\xi(\mathbf{y}) = \sum_{\alpha \in \mathbb{N}^n} \xi^\alpha \frac{\boldsymbol{y}^\alpha}{\alpha!} = e^{\xi_1 \boldsymbol{y}_1 + \cdots + \xi_n \boldsymbol{y}_n} = e^{\langle \xi, \boldsymbol{y} \rangle},$$

that is, an exponential series in the basis $(\boldsymbol{y}^\alpha)_{\alpha \in \mathbb{N}^n}$.

Definition 2.4 (Polynomial-Exponential series).

$$\mathcal{P}ol\mathcal{E}xp(\mathbf{y}) = \left\{ \Lambda(\mathbf{y}) = \sum_{i=1}^{r} \omega_i(\mathbf{y})\, \mathbf{e}_{\xi_i}(\mathbf{y}) \mid \omega_i(\mathbf{y}) \in \mathbb{K}[\mathbf{y}], \xi_i \in \mathbb{K}^n \right\}$$

where $\mathbf{e}_{\xi_i}(\mathbf{y})$ is *the series representing the evaluation* $\mathbf{e}_{\xi_i} : p \in R \mapsto p(\xi_i)$ *at* ξ_i.

For an ideal $I \subset R$, we denote by $I^\perp \subset R^*$ the space of linear forms $\Lambda \in R^*$, such that $\forall p \in I$, $\langle \Lambda \mid p \rangle = 0$. Similarly, for a vector space $D \subset \mathbb{K}[[\mathbf{y}]]$, we denoted by $D^\perp \subset \mathbb{K}[\mathbf{x}]$ the space of polynomials $p \in \mathbb{K}[\mathbf{x}]$, such that $\forall \Lambda \in D$, $\langle \Lambda \mid p \rangle = 0$.
The dual space R^* has a natural structure of R-module, defined as follows: $\forall \Lambda \in R^*, \forall p, q \in R$,

$$\langle p \star \Lambda \mid q \rangle = \langle \Lambda \mid pq \rangle.$$

We verify that x_i acts on the series in $\mathbf{y}$ by derivation: $\forall \Lambda \in \mathbb{K}[[\mathbf{y}]], x_i \star \Lambda = \frac{\partial}{\partial_{y_i}}(\Lambda)$.

Definition 2.5 (Inverse system). *For* $\boldsymbol{\Lambda} \subset R^*$, *the* inverse system *spanned by* $\boldsymbol{\Lambda}$ *is*

$$\langle\langle \boldsymbol{\Lambda} \rangle\rangle = \langle p \star \Lambda; p \in R, \Lambda \in \boldsymbol{\Lambda} \rangle$$

As x_i acts as a derivation on the series $\in \mathbb{K}[\mathbf{y}]]$, the inverse system generated by $\boldsymbol{\Lambda}$ is represented by the vector space spanned by all possible derivations $\partial^\alpha_{\mathbf{y}}(\Lambda(\mathbf{y}))$ for all $\alpha \in \mathbb{N}^n$, $\Lambda \in \boldsymbol{\Lambda}$. If $\Lambda(\mathbf{y}) \in \mathcal{P}ol\mathcal{E}xp(\mathbf{y})$, we verify that $\langle\langle \Lambda \rangle\rangle$ is finite dimensional. See [30].

Theorem 2.6. *Assume that* $\mathbb{K} = \overline{\mathbb{K}}$ *is algebraically closed and let* $\mathcal{A} = \mathcal{R}/I$ *be an Artinian algebra with* $\mathcal{V}(I) = \{\xi_1, \ldots, \xi_{r'}\}$ *and* $I = Q_1 \cap \cdots \cap Q_{r'}$ *where* Q_i *are the* $\mathbf{m}_{\xi_i}$*-primary components of* I. *Then,*

$$\mathcal{A}^* = \oplus_{i=1}^{r'} \mathcal{D}_i\, \mathbf{e}_{\xi_i}(\mathbf{y}) \subset \mathcal{P}ol\mathcal{E}xp(\mathbf{y})$$

- $\mathcal{D}_i = \mathcal{A}_i^* = \langle\langle \omega_{i,1}(\mathbf{y}), \ldots, \omega_{i,l_i}(\mathbf{y}) \rangle\rangle$ *with* $\omega_{i,j}(\mathbf{y}) \in \mathbb{K}[\mathbf{y}]$.
- $\dim_{\mathbb{K}}(\mathcal{D}_i) = \mu_i$ *multiplicity of* ξ_i.

where $\mathcal{D}_i$ *is the* inverse system *generated by* $\omega_{i,1}(\mathbf{y}), \ldots, \omega_{i,l_i}(\mathbf{y}) \in \mathbb{K}[\mathbf{y}]$

$$\langle\langle \omega_{i,1}(\mathbf{y}), \ldots, \omega_{i,l_i}(\mathbf{y}) \rangle\rangle = \langle \partial^\alpha_{\mathbf{y}}(\omega_{i,j}), \alpha \in \mathbb{N}^n, j = 1, \ldots, l_i \rangle$$

For symbolic-numeric algorithms computing the inverse system $\mathcal{D}_i = \mathcal{A}_i^* = Q_i^\perp$ at ξ_i from the generators of I, see [28,18,26,25] and Section 3.1.

3 Computing effectively these structures

From a computational point of view, we handle the algebraic structure of an Artinian algebra, via so-called *Truncated Normal Forms*:

Definition 3.1 (Truncated Normal Form). *Let* $W \subset R$ *and* V *be a* $\mathbb{K}$*-vector space. A* Truncated Normal Form (TNF) *for the ideal* I *from* W *to* V *is a linear map* $\mathbf{N} : W \to V$ *such that the sequence*

$$0 \longrightarrow K \longrightarrow W \xrightarrow{\mathbf{N}} V \longrightarrow 0$$

is exact, $I = (K)$ *is the ideal of* R *generated by* $K := \ker \mathbf{N}$, $I \cap W = K$ *and* $I + W = R$.

For other characterisations of TNF, see e.g. [36,32]. By definition, if N is a TNF, then it induces an isomorphism between $\mathcal{A} = R/(\ker \mathbf{N})$ and V. For any set $B \subset R$, let $B^+ = B \cup x_1 \cdot B \cup \cdots \cup x_n \cdot B$.
The multiplicative structure of an Artinian algebra $\mathcal{A} = R/I$ can be recovered using the following result:

Proposition 3.2. *Let* $\mathbf{N} : W \to V$ *be a TNF such that there exists* $B \subset W$ *with* $B^+ \subset W$ *and* $\mathbf{N}_{|B} : B \to V$ *bijective. Let* $I = (\ker \mathbf{N})$. *Then*

$$\begin{aligned} \mathbf{M}_i : B &\to B \\ b &\mapsto (\mathbf{N}_{|B})^{-1} \circ \mathbf{N}(x_i b). \end{aligned}$$

is the operator of multiplication by x_i *in the quotient* R/I.

Hereafter is the general form of the eigensolver algorithm, that is implemented in AlgebraicSolvers.jl:

Algorithm 1 EIGENSOLVE

1: **function** EIGENSOLVE($\mathbf{P}$)
2: **Input**: $\mathbf{P} = (P_1, \ldots, P_m) \in R^m$
3: Compute a Truncated Normal Form $N : W \longrightarrow V$ for $I = (P_1, \ldots, P_m)$;
4: Extract a basis $B \subset W$ such that $N_{|B}$ is bijective and $B^+ \subset W$;
5: Build multiplication matrices $M = \{M_{x_1}, \ldots, M_{x_n}\}$ in the basis B from N;
6: Compute the eigenvalues $\boldsymbol{\lambda}$ of a random combination M_{rnd} of $M_1, \ldots, M_n$ and their multiplicities;
7: **if** the eigenvalues are simple **then**
8: Compute the points Ξ by joint diagonalisation of M.
9: **else**
10: Compute the points Ξ and their multiplicities $\boldsymbol{\mu}$ by a joint Schur factorization of M.
11: **end if**
12: **Output**: the roots Ξ and their multiplicities $\boldsymbol{\mu}$.
13: **end function**

Determining the multiplicity of the eigenvalues $\boldsymbol{\lambda}$ (step 6) is performed by a clustering function that uses a threshold ε (`1.e-5` by default) to identify points in the same cluster.
The joint diagonalization (step 8) is implemented as an iterative process, optimizing the sum of the square off-diagonal norm of the matrices $E^{-1} M_i E$ with respect to the (unknown) eigenvector matrix E, starting from the eigenvector matrix of M_{rnd} in Algorithm 1 (see [37]).
The joint Schur factorization (step 10) applies the Schur Factorization of M_{rnd} to $M_1, \ldots, M_n$ and uses the result of the clustering function to deduce the operators of multiplication by the variables x_i in the local algebras associated to the roots. Their coordinates are recovered from the trace of these local multiplications (see Proposition 2.3). Their multiplicities $\boldsymbol{\mu}$ are the sizes of the clusters.
Hereafter, we illustrate these techniques and their implementation in the package AlgebraicSolvers.jl[1] on few examples.

3.1 Solving polynomial systems

In this section, we illustrate the computation of roots from eigen computations, in the presence of multiple roots. We consider the following system of two equations in two variables x_1, x_2:

```
using DynamicPolynomials, LinearAlgebra, AlgebraicSolvers, Groebner
x = (@polyvar x[1:2])[1]
P = [x[1]^2 + x[1] - x[2], x[2]^2 + x[1] - x[2]]
```

```
-x_2 + x_1 + x_1^2
-x_2 + x_1 + x_2^2
```

We compute a TNF via Grobner basis computation with exact coefficients (using the package `Groebner.jl` [12]):

[1] See also its documentation at https://algebraicgeometricmodeling.github.io/AlgebraicSolvers.jl/.

```
GB = Grobner(
    Groebner.DegRevLex,      # function of the variables defining the monomial ordering
    Groebner.groebner,       # function for computing Grobner basis
    Groebner.normalform,     # function that computes the normal form of a polynomial
    Groebner.quotient_basis  # function that computes a basis of the quotient algebra
)
N, L = tnf(P, GB); round.(N, digits=6)
```

```
4×8 Matrix{Float64}:
 1.0  0.0  0.0  0.0   0.0   0.0   0.0   0.0
 0.0  1.0  0.0  0.0   1.0   1.0  -1.0   1.0
 0.0  0.0  1.0  0.0  -1.0  -1.0   1.0  -1.0
 0.0  0.0  0.0  1.0   0.0   0.0   1.0  -1.0
```

Let $B = [1, x_1, x_2, x_1x_2]$ be the first 4 elements of L. Then we verify that $L = B^+$:

```
(1, x_2, x_1, x_1*x_2, x_2^2, x_1^2, x_1*x_2^2, x_1^2*x_2)
```

The matrix `N` represents the TNF map $N : W \longrightarrow \mathbb{R}^4$ where $W = \langle L \rangle$. It is computed as follows:

- first we compute a Grobner basis `G` of `P`, using the function `Groebner.groebner` with the monomial ordering `Groebner.DegRevLex(variables(P))`. The coefficients of the polynomials in this computation are rational $\in \mathbb{Q}$;
- then we compute a basis `B` of the quotient by the ideal $I = (P_1, P_2)$ using `Groebner.quotient_basis`;
- Finally we compute the reduction of the monomials in L by `G` (using `Groebner.normalform`) and decompose the remainder in the basis `B` using `Float64` numbers. This gives the columns of `N`.

We compute the multiplication matrices by the variables x_1, x_2 in the basis B, using the TNF `N` indexed by the monomials L:

```
M = mult_matrices(N, L , collect(1:4), x); round.(M[1], digits=6)
```

```
4×4 Matrix{Float64}:
 0.0  0.0   0.0   0.0
 0.0  0.0   1.0   1.0
 1.0  0.0  -1.0  -1.0
 0.0  1.0   0.0  -1.0
```

To obtain the roots and their multiplicities, we perform a joint Schur factorization:

```
Xi, ms, Z = schur_dcp(M); round.(Xi, digits=6)
```

```
2×2 Matrix{ComplexF64}:
 -0.0+0.0im  -2.0+0.0im
 -0.0+0.0im   2.0+0.0im
```

The solutions, which can be obtained directly using the function `solve(P,GB)`, are the columns of the matrix `Xi`. The corresponding multiplicities are `length.(ms)`:

```
 3  1
```

and the Schur factor is `Z`.
Instead of using Grobner basis computation to get a TNF, we could also have used a resultant matrix construction (hereafter Macaulay construction):

```
R, L = res_matrix(P, Macaulay()); round.(R, digits=6)
```

```
6×10 SparseArrays.SparseMatrixCSC{Float64, Int64} with 18 stored entries:
  .   -1.0  1.0    .     .   1.0   .    .    .    .
  .     .    .   -1.0   1.0   .    .    .   1.0   .
  .     .    .     .   -1.0  1.0   .    .    .   1.0
  .   -1.0  1.0   1.0    .    .    .    .    .    .
  .     .    .   -1.0   1.0   .   1.0   .    .    .
  .     .    .     .   -1.0  1.0   .   1.0   .    .
```

The rows represent the monomial multiples of degree ≤ 3 of the equations `P[i]`. It is a matrix of size 6×10, which columns are indexed by the monomials:

```
(1, x_2, x_1, x_2^2, x_1*x_2, x_1^2, x_2^3, x_1*x_2^2, x_1^2*x_2, x_1^3)
```

The corresponding TNF, using the function `tnf(P,Macaulay())`, is the kernel of `R`:

```
4×10 Matrix{Float64}:
 1.0  0.0  0.0  -0.0  0.0  -0.0  -0.0  -0.0  -0.0  -0.0
 0.0  0.0  0.0  -0.0  1.0  -0.0  -1.0   1.0  -1.0   1.0
 0.0  0.0  1.0  -1.0  0.0  -1.0  -1.0   1.0  -1.0   1.0
 0.0  1.0  0.0   1.0  0.0   1.0   1.0  -1.0   1.0  -1.0
```

Using this TNF yields the same basis B and a numerical accuracy on the roots of the same order as the one obtained via Grobner basis computation.
We analyze now the inverse system of the multiple point $\xi_0 = [0, 0]$:

```
D, B0 = invsys(P,[0,0]); D
```

```
1
dx_1 + dx_2
dx_1^2 + dx_2 + dx_1*dx_2 + dx_2^2
```

These series represent a basis of the inverse system of $I = (P_1, P_2)$ at the point ξ_0. They are described as series (`AlgebraicSolvers.Series`), in terms the dual basis $dx_1^{\alpha_1} dx_2^{\alpha_2}$ of the monomial basis $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2}$.
A basis of the local algebra $\mathcal{A}_{\xi_0}$ is `B0 =` $(1, x_1, x_1^2)$. We verify that `B0` is dual to `D`. This inverse system is generated by `D[3]`, since the following series span the same space of series as `D`:

```
x[1]^2*D[3], x[2]^2*D[3], x[1]*x[2]*D[3], x[1]*D[3], x[2]*D[3], D[3]
```

```
(1, 1, 1, dx_1 + dx_2, 1 + dx_1 + dx_2, dx_1^2 + dx_2 + dx_1*dx_2 + dx_2^2)
```

3.2 Decomposing symmetric tensors

The *Generalized Additive Decomposition* (GAD) of a symmetric tensor, or equivalently a homogeneous polynomial (or form) F of degree d in the variables $\mathbf{x} = (x_0, \ldots, x_n)$ is a decomposition of the form

$$F = \sum_{i=1}^{r} \omega_i(\mathbf{x}) \, (\xi_i \cdot \mathbf{x})^{d-k_i} \tag{2}$$

where $\omega_i(\mathbf{x})$ is a homogeneous polynomial in the variables $\mathbf{x}$ of degree k_i with $0 \leq k_i \leq d$, and $\xi_i \in \mathbb{C}^{n+1}$. When all the degrees k_i vanish, such a decomposition is called a *Waring decomposition* and the minimal r in a Waring decomposition is called the *rank* of F. See [2] for more details on the GAD algorithm.

```
using DynamicPolynomials, LinearAlgebra, AlgebraicSolvers, TensorDec
X  = (@polyvar x[0:2])[1]; x = X[2:end]
F = (X[1]-2*X[2]+ 2*X[3])^5 + (X[1]*X[3]+ X[2]^2+2*X[2]*X[3]+ X[3]^2)*X[1]^3
```

$$32x_2^5 - 160x_1x_2^4 + 320x_1^2x_2^3 - 320x_1^3x_2^2 + 160x_1^4x_2 - 32x_1^5 + 80x_0x_2^4 - 320x_0x_1x_2^3 + 480x_0x_1^2x_2^2 - 320x_0x_1^3x_2 + 80x_0x_1^4 + 80x_0^2x_2^3 - 240x_0^2x_1x_2^2 + 240x_0^2x_1^2x_2 - 80x_0^2x_1^3 + 41x_0^3x_2^2 - 78x_0^3x_1x_2 + 41x_0^3x_1^2 + 11x_0^4x_2 - 10x_0^4x_1 + x_0^5$$

We compute the series associated with the tensor `F` in degree d by apolar duality:

```
sigma = apolar_dual(F)
```

```
32dx_2^5 - 32dx_1*dx_2^4 + 32dx_1^2dx_2^3 - 32dx_1^3dx_2^2 + 32dx_1^4dx_2 - 32dx_1^5 + 16
    dx_0*dx_2^4 - 16dx_0*dx_1*dx_2^3 + 16dx_0*dx_1^2*dx_2^2 - 16dx_0*dx_1^3*dx_2 + 16
    dx_0*dx_1^4 + 8dx_0^2dx_2^3 - 8dx_0^2dx_1*dx_2^2 + 8dx_0^2dx_1^2dx_2 - 8dx_0^2dx_1^3
     + 410dx_0^3dx_2^2 - 390dx_0^3dx_1*dx_2 + 410dx_0^3dx_1^2 + 11//5dx_0^4dx_2 - 2dx_0
    ^4dx_1 + dx_0^5
```

where dx^α are the linear functionals $\mathbf{y}^{[\alpha]} = \frac{\mathbf{y}^\alpha}{\alpha!}$ dual to the monomial basis $\mathbf{x}^\alpha$. We deduce the Hankel (or Catalecticant) matrix in degree $2, 3$. From this, by Theorem 3.9 of [31] we recover the TNF `N` by extracting the submatrix corresponding to the rows indexed by the monomial basis $B = (1, x_1, x_2, x_1x_2)$, which, in the homogenized setting, corresponds to $(x_0^2, x_0x_1, x_0x_2, x_1x_2)$.

```
L2 = reverse(monomials(X,2)); L3 = reverse(monomials(X,3))
Hk = hankel(sigma,L2,L3)
I = [1,2,3,5]; N = Hk[I,:]
```

```
4×10 Matrix{Rational{Int64}}:
    1       -2      11//5   41//10  -39//10  41//10   -8     8   -8     8
   -2      41//10  -39//10   -8        8      -8       16   -16   16   -16
  11//5   -39//10   41//10    8       -8       8      -16    16  -16    16
 -39//10    8        -8     -16       16     -16       32   -32   32   -32
```

At this point we can compute the multiplication matrices by the variables x_1, x_2, while substituting x_0 by 1:

```
M = mult_matrices(N, subs.(L3, X[1] => 1), I, x)
```

```
2-element Vector{Matrix{Rational{Int64}}}:
 [0 0 0 0; 1 -2//3 0 -2//3; 0 2//3 0 2//3; 0 -1//3 1 -4//3]
 [0 0 0 0; 0 0 -2//3 2//3; 1 0 2//3 -2//3; 0 1 -1//3 4//3]
```

We recover the points and their multiplicities by performing a joint Schur factorization.

```
Xi, ms = schur_dcp(M, 1.e-3); Xi = vcat(ones(1, size(Xi,2)), Xi)
```

```
3×2 Matrix{ComplexF64}:
  1.0+0.0im  1.0+0.0im
 -2.0+0.0im  0.0+0.0im
  2.0+0.0im  0.0+0.0im
```

We may assume, without loss of generality, that the points do not lie in the hyperplane $\{x_0 = 0\}$. Indeed, over an infinite field, a generic linear change of coordinates ensures that the first coordinate of each point ξ_i is nonzero. After rescaling, we may further assume that these first coordinates are all equal to 1. We have added a first row of 1's to obtain the corresponding projective points. The linear forms in the GAD decomposition (2) are $L_i = (\xi_i \cdot \mathbf{x})$ where $\xi_i =$`Xi[:,i]` is the i$^{\text{th}}$ column of `Xi`.
We are left with computing k, the vector of degrees of the homogeneous polynomials ω_i's. We compute it from the nil-indices of the multiplication operators in the local algebras $\mathcal{A}_{\xi_i}$ (see [2]) and get

```
(0, 2)
```

To complete the decomposition, we compute the coefficients of the ω_i's, by solving a Vandermonde-like linear system in the least-square sense (see [2]) and get `W`:

```
((1.0 + 0.0im), x_2^2 + (2.0 + 0.0im)x_1*x_2 + x_1^2 + x_0*x_2)
```

These weights represent the generators of the inverse systems of the local algebras associated with F. We can check that the full Artinian algebra associated with F is isomorphic to the one defined in Section 3.1, by comparing the points and the generators of their inverse systems.
We verify the accuracy of the reconstruction by forming the reconstructed tensor from the linear forms L and the weights `W` of degree k_i.

```
T = sum(W[i]*L[i]^(maxdegree(F)-k[i]) for i in 1:length(W))
e = apolar_norm(F-T)
```

```
9.342121376789858e-14
```

The reconstruction error is of order 10^{-13}, demonstrating the accuracy of the reconstruction.

References

1. W. Auzinger and H. J. Stetter, *An Elimination Algorithm for the Computation of All Zeros of a System of Multivariate Polynomial Equations*, Birkhäuser Basel, 1988, p. 11–30.
2. E. Barrilli, B. Mourrain, and D. Taufer, *Generalized additive decompositions of symmetric tensors*. preprint, hal-05339397, Oct. 2025.
3. A. Bernardi, J. Brachat, P. Comon, and B. Mourrain, *General tensor decomposition, moment matrices and applications*, Journal of Symbolic Computation, 52 (2013), pp. 51–71.
4. J. Berthomieu, C. Eder, and M. Safey El Din, *msolve: A Library for Solving Polynomial Systems*, in 2021 International Symposium on Symbolic and Algebraic Computation, 46th International Symposium on Symbolic and Algebraic Computation, Saint Petersburg, Russia, July 2021, ACM, pp. 51–58.
5. J. Brachat, P. Comon, B. Mourrain, and E. Tsigaridas, *Symmetric tensor decomposition*, Linear Algebra and its Applications, 433 (2010), pp. 1851–1872.
6. A. Bunse-Gerstner, R. Byers, and V. Mehrmann, *Numerical methods for simultaneous diagonalization*, SIAM journal on matrix analysis and applications, 14 (1993), pp. 927–949.
7. J.-F. Cardoso and A. Souloumiac, *Jacobi angles for simultaneous diagonalization*, SIAM journal on matrix analysis and applications, 17 (1996), pp. 161–164.
8. 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, ISSAC '97, ACM Press, 1997, pp. 133–140.
9. D. A. Cox, *Applications of Polynomial Systems*, vol. 134 of Regional conference series in mathematics, American Mathematical Society, Providence, Rhode Island, 2020. With contributions by Carlos D'Andrea, Alicia Dickenstein, Jonathan Hauenstein, Hal Schenck, Jessica Sidman.
10. D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Undergraduate Texts in Mathematics, Springer, 1992.
11. J.-P. Dedieu, *Newton-Raphson Method*, Springer Berlin Heidelberg, 2015.
12. A. Demin and S. Gowda, *Groebner.jl: A package for Gröbner bases computations in Julia*, preprint arXiv:2304.06935, (2023).
13. B. Dumitrescu, *Positive Trigonometric Polynomials and Signal Processing Applications*, Springer International Publishing, 2017.
14. C. Eder and J.-C. Faugère, *A survey on signature-based algorithms for computing gröbner bases*, Journal of Symbolic Computation, 80 (2017), p. 719–784.
15. M. Elkadi and B. Mourrain, *Introduction à la résolution des systèmes polynomiaux*, Springer Berlin Heidelberg, 2007.
16. I. Z. Emiris and B. Mourrain, *Matrices in elimination theory*, Journal of Symbolic Computation, 28 (1999), p. 3–44.
17. J.-C. Faugère, *FGb: A Library for Computing Gröbner Bases*, Springer Berlin Heidelberg, 2010, p. 84–87.
18. J. D. Hauenstein, B. Mourrain, and A. Szanto, *On deflation and multiplicity structure*, Journal of Symbolic Computation, 83 (2016), pp. 228–253.

19. D. Henrion and J.-B. Lasserre, *Detecting Global Optimality and Extracting Solutions in GloptiPoly*, Springer Berlin Heidelberg, Aug. 2005, p. 293–310.
20. A. Iarrobino and V. Kanev, *Power Sums, Gorenstein Algebras, and Determinantal Loci*, Springer Berlin Heidelberg, 1999.
21. M. Kreuzer and L. Robbiano, *Deformations of border bases*, Collectanea mathematica, 59 (2008), p. 275–297.
22. Z. Kukelova, M. Bujnak, and T. Pajdla, *Automatic Generator of Minimal Problem Solvers*, Springer Berlin Heidelberg, 2008, p. 302–315.
23. J. B. Lasserre, *Moments, Positive Polynomials and Their Applications*, Imerial College Press, 2009.
24. F. S. Macaulay, *The Algebraic Theory of Modular Systems*, Cambridge University Press, 1916.
25. A. Mantzaflaris, B. Mourrain, and A. Szanto, *Punctual Hilbert Schemes and Certified Approximate Singularities*, in ISSAC 2020 - International Symposium on Symbolic and Algebraic Computation, Kalamata, Greece, 2020.
26. ———, *A certified iterative method for isolated singular roots*, Journal of Symbolic Computation, 115 (2022), pp. 223–247.
27. H. M. Möller and H. J. Stetter, *Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems*, Numerische Mathematik, 70 (1995), pp. 311–329.
28. B. Mourrain, *Computing the isolated roots by matrix methods*, Journal of Symbolic Computation, 26 (1998), p. 715–738.
29. ———, *A New Criterion for Normal Form Algorithms*, Springer Berlin Heidelberg, 1999, p. 430–442.
30. B. Mourrain, *Polynomial–exponential decomposition from moments*, Foundations of Computational Mathematics, 18 (2017), p. 1435–1492.
31. ———, *Truncated Normal Forms for Solving Algebraic Equations*. preprint, hal-05135952, June 2025.
32. B. Mourrain, S. Telen, and M. Van Barel, *Truncated normal forms for solving polynomial systems: Generalized and efficient algorithms*, Journal of Symbolic Computation, 102 (2021), p. 63–85.
33. B. Mourrain and P. Trebuchet, *Generalized normal forms and polynomial system solving*, in Proceedings of the 2005 international symposium on Symbolic and algebraic computation, ISSAC05, ACM, 2005, p. 253–260.
34. A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, WORLD SCIENTIFIC, 2005.
35. H. J. Stetter, *Numerical Polynomial Algebra*, Society for Industrial and Applied Mathematics, Jan. 2004.
36. S. Telen, B. Mourrain, and M. V. Barel, *Solving polynomial systems via truncated normal forms*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), p. 1421–1447.
37. J. van der Hoeven and B. Mourrain, *Efficient certication of numeric solutions to eigenproblems*, in Lect. Notes in Computer Science, Blömer, J., Kotsireas, I. S., Kutsia, T., Simos, and D. E., eds., vol. 10693 of Lect. Notes in Computer Science, Vienna, Austria, Nov. 2017, Springer International Publishing, pp. 81–94.

Acknowlegdment. This work has been supported by European Union's HORIZON–MSCA-2023-DN-JD programme under the Horizon Europe (HORIZON) Marie Skłodowska-Curie Actions, grant agreement 101120296 (TENORS).