EFFICIENT KRYLOV-BASED METHODS FOR SOLVING OVERDETERMINED POLYNOMIAL SYSTEMS*

NITHIN GOVINDARAJAN†, NICO VERVLIET†, RAPHAËL WIDDERSHOVEN†, AND LIEVEN DE LATHAUWER†

Abstract. Computing the null space of resultant matrices, such as Macaulay matrices, is a central step in many numerical methods for solving systems of polynomial equations. For overdetermined systems, this step becomes the dominant computational bottleneck, even when the number of solutions is small. We apply two Krylov-based algorithms—LSQR and the augmented implicitly restarted Lanczos bidiagonalization (IRLBA) method—to compute numerical bases for the approximate null spaces without ever forming explicit factorizations of the Macaulay matrices themselves. To enable efficient iterations, we develop fast matrix–vector product algorithms based on Fourier and cosine transforms, exploiting the multi-level Toeplitz structure of Macaulay matrices in the monomial basis and the Toeplitz-plus-Hankel structure in the Chebyshev basis. Theoretical complexity estimates and numerical experiments indicate that, when combined with a tensor-based approach for recovering roots from partially computed null spaces, these methods can substantially reduce memory usage and floating-point operations compared to the standard SVD method.

Key words. Overdetermined polynomial systems, Macaulay matrices, canonical polyadic decomposition, implicitly restarted Lanczos bidiagonalization, LSQR

AMS subject classifications. 65F10, 65F15, 15A69

1. Introduction.

The problem of solving systems of polynomial equations arises in a wide range of scientific disciplines such as biology, kinematics, robotics, and statistics [14,33,43]. Over the years, computational algebraic geometry has developed a variety of powerful tools to tackle such problems [16, 17, 18, 46]. In the case where the system admits only isolated solutions, a prominent strategy—alongside numerical methods such as homotopy continuation [31]—is to reformulate the task of finding roots as an eigenvalue problem [15, 44], or, more recently, as a tensor decomposition problem [49, 50]. The central idea behind the algebraic approach is to construct multiplication matrices that encode the structure of the quotient algebra associated with the ideal generated by the system. These matrices can be obtained by various means, either symbolically using Gröbner bases [2, 10], or numerically using linear algebra techniques. The methods developed in [5, 9, 12, 13, 27, 29, 35, 36, 38, 45, 47] all fall within this paradigm and trace their theoretical foundations back to resultant theory and commutative algebra [4, 21, 22].

Algebraic methods provide an elegant mechanism to retrieve all solutions of a polynomial system simultaneously. However, their practical efficiency depends strongly on the system's size and structure. In certain fully numerical frameworks [9, 20, 50], this complexity curse manifests itself in the form of computing a null space basis of a very large resultant matrix, followed by solving an eigenvalue or tensor decomposition problem of a prohibitively large size. For square systems, where the number of equations S equals the number of variables N, the exponential blow-up in complexity is largely intrinsic to the problem. Since the number of solutions R

*Submitted to the editors June 29, 2026.

Funding: This work was funded by (1) the Flemish Government: this research received funding under the AI Research Program. LDL, NG and RW are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. (2) KU Leuven Internal Funds: iBOF/23/064, C14/22/096. NV holds a research grant from the Research Foundation Flanders (FWO) (12ZM220N).

†E. E. Dept. (ESAT), KU Leuven, Leuven, Belgium (nithin.govindarajan@kuleuven.be, nico.vervliet@kuleuven.be, raphael.widdershoven@kuleuven.be, lieven.delathauwer@kuleuven.be)

increases exponentially with N, a lower bound is imposed on the computational cost to retrieve all solutions of the system. For overdetermined systems ($S > N$), the situation is more subtle. The number of solutions can vary widely in an overdetermined system, from no solutions to the maximal bounds predicted by Bézout's and Bernstein–Kushnirenko's results [23, 40]. Consequently, when R is small, the second step of recovering the roots becomes relatively inexpensive. This characteristic makes overdetermined systems appear more tractable than their square counterparts. Unfortunately, one must still contend with the primary computational challenge of extracting the relevant null space from a very large matrix in the first place.

Significant computational progress in solving overdetermined polynomial systems hinges on addressing the bottleneck of extracting the null space of a resultant matrix. Resultant matrices, such as those of the Macaulay-type, possess rich algebraic structures such as sparsity, multi-level convolutional patterns, and recursive properties. Several studies have focused on exploiting these structures for more efficient null space computation [8, 25, 37, 51, 54]. These methods have achieved substantial computational gains over the baseline SVD approach within their respective regimes. However, despite their advances, the methods remain suboptimal for overdetermined systems, as none fully leverage the fact that the null space dimension is small whenever the root count is small. Ideally, memory usage should scale with the storage required to represent an R-dimensional subspace; however, existing algorithms often require constructing considerably larger intermediate matrices before extracting a null space basis. In this paper, we propose two approaches for computing a null space basis whose memory requirements scale more favorably with the null space dimension.

The proposed methods are built on two well-established Krylov subspace frameworks for solving least-squares problems and computing extremal singular vectors. The first approach employs the LSQR algorithm [39] with a zero right-hand side to obtain null vectors, while the second uses the augmented implicitly restarted block Lanczos bidiagonalization (IRLBA) algorithm [6] to compute right singular vectors associated with the smallest singular values. Both approaches are rooted in the Golub–Kahan bidiagonalization process, which is a natural choice for extracting fundamental matrix subspaces of large matrices. A key advantage of LSQR and IRLBA is that they avoid explicit factorization or the direct solution of linear systems, requiring only matrix-vector products. In contrast, shift-and-invert techniques—such as those used in `svds` (ARPACK) [30]—while effective for rapidly converging to the smallest singular triplets, rely on explicit matrix factorizations that are infeasible in our setting, as the multi-level structures of the matrices make direct factorization or solution prohibitively expensive.

1.1. Contributions. In this paper, we demonstrate the effectiveness of the proposed Krylov methods for extracting the null space in overdetermined polynomial systems. We focus on systems whose monomial support lies on a rectilinear grid. In the bivariate case, this corresponds to considering generic polynomials of multi-degree (d_1, d_2) rather than total degree D, i.e.,

$$f(x,y) = \sum_{k=0}^{d_1}\sum_{l=0}^{d_2} c_{kl}x^k y^l \quad \text{instead of} \quad f(x,y) = \sum_{0\leq k+l\leq D} c_{kl}x^k y^l.$$

The main contributions of this work are:

1. *Generalization of the tensor-based Macaulay method* [49, 50] to polynomial systems on rectilinear grids expressed in arbitrary bases defined by a three-term recurrence relation. While modified homogenization procedures can

handle such systems to avoid spurious roots at infinity [9, 47], the tensor framework offers two unique perspectives:
   (a) It enables recovery of roots without constructing multiplication tables by leveraging the shift properties of Vandermonde vectors and relying solely on low-rank structure.
   (b) Under mild conditions, a single null space vector often suffices to recover all roots, thereby reducing the computational burden of the initial null space computation and the associated cost of the Krylov methods.
2. *Computationally efficient Krylov methods* that leverage fast matrix-vector products exploiting the multi-level Toeplitz and multi-level Toeplitz-plus-Hankel structure of Macaulay matrices in the monomial and Chebyshev bases, respectively. Specifically, if d denotes the degree of the polynomials in each coordinate and $D \gg d$ the Macaulay degree, our methods produce a null space vector in $\mathcal{O}(SND^{2N} \log D)$ floating point operations (flop) and equires storing $\mathcal{O}(Sd^{2N} + D^N)$ floats using fast Fourier and cosine transforms. In contrast, standard SVD-based approaches require $\mathcal{O}(SD^{3N})$ flop and $\mathcal{O}(SD^{2N})$ memory. See Tables 1 and 2 for a detailed comparison.

1.2. Outline. Section 2 generalizes the tensor-based Macaulay method to polynomial systems with rectilinear monomial support in arbitrary bases. Section 3 develops fast matrix–vector product algorithms for the Macaulay matrices in the monomial and Chebyshev bases. Section 4 introduces and analyzes two Krylov-based null space methods using LSQR and IRLBA. Section 5 presents numerical experiments illustrating the practical efficiency of the proposed approach, and section 6 concludes with a summary and perspectives for future work.

1.3. Notation and review of basic mathematical concepts.

1.3.1. (Multi-)linear algebra. Scalars, vectors, matrices and higher-order tensors are expressed with regular lower or upper-case (Greek) letters, bold lower-case, bold capitals, and calligraphic letters, respectively, i.e., a, $\boldsymbol{a}$, $\mathbf{A}$, and $\mathcal{A}$. At times, we shall use multi-indexed subscript notation, e.g., $[\mathcal{A}]_{\boldsymbol{i}:\boldsymbol{j}}$ is short-hand for $[\mathcal{A}]_{i_1:j_1,\ldots,i_N:j_N}$. A vector of n zeros is denoted by $\mathbf{0}_n$. The n-by-n identity matrix is denoted by $\mathbf{I}_n$, whereas $\mathbf{I}_{m,n}$ describes the m-by-n matrix with ones on the main diagonal and zeros elsewhere. Diagonal matrices are denoted using the $\operatorname{diag}(\cdot)$ notation. The Kronecker product between two matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted with the symbol $\otimes$. Let $\|\boldsymbol{v}\|_p := (\sum_{i=1}^n |v_i|^p)^{1/p}$ and $\|\mathbf{A}\|_{\mathrm{F}} := \sqrt{\sum |a_{ij}|^2}$. The rank of a tensor, denoted $\operatorname{rank} \mathcal{A}$, is the minimal number of outer product terms whose sum represents $\mathcal{A}$. The column and null spaces of $\mathbf{A}$ are denoted with $\operatorname{col} \mathbf{A}$ and $\operatorname{null} \mathbf{A}$, respectively. The symbols $(\cdot)^\top$ and $(\cdot)^*$, are used to denote transpose and conjugate transpose. We use $\mathcal{T} \cdot_n \mathbf{X}$ to denote the mode-n product, i.e., $[\mathcal{T} \cdot_n \mathbf{X}]_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n} [\mathcal{T}]_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} [X]_{j i_n}$. The Hadamard (i.e., element-wise) product between two tensors of similar shape is denoted with $*$. Polyadic decompositions of a tensor are conveniently expressed using bracket notation, e.g., $[\![\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3]\!] := \sum_{r=1}^R [\mathbf{U}_1]_{:,r} \otimes [\mathbf{U}_2]_{:,r} \otimes [\mathbf{U}_3]_{:,r}$, where $\otimes$ denotes the outer product. A polyadic decomposition is canonical (CPD) if it has minimal rank. A CPD is essentially unique if any other decomposition can only differ by scaling/counterscaling within terms and permutation of terms [41]. Krylov subspaces are denoted by $\mathscr{K}_k(\mathbf{A}, \boldsymbol{x}) := \operatorname{span}\{\boldsymbol{x}, \mathbf{A}\boldsymbol{x}, \ldots, \mathbf{A}^{k-1}\boldsymbol{x}\}$.

1.3.2. Polynomials. Let $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_N^{\alpha_N}$ denote a monomial over the variables $\boldsymbol{x} = (x_1, \ldots, x_N)$ with exponent $\boldsymbol{\alpha} \in \mathbb{N}^N$. The monomial $\boldsymbol{x}^{\boldsymbol{\alpha}}$ has *total* degree $\deg(\boldsymbol{x}^{\boldsymbol{\alpha}}) := \alpha_1 + \ldots + \alpha_N$. $\boldsymbol{x}^{\boldsymbol{\alpha}}$ may also be viewed in terms of multi-degree.

This is done by introducing a partitioning $\{\mathscr{C}_k\}_{k=1}^K$ of the set $\{1,\dots,N\}$, with $\mathscr{C}_k = \{\zeta_{k,1},\dots,\zeta_{k,N_k}\}$ and $N_1+\ldots+N_K=N$, so that we may "split" $\boldsymbol{x}$ into $1 \leq K \leq N$ disjoint variable groups $\mathscr{P} = (\boldsymbol{\xi}_k)_{k=1}^K$, where $\boldsymbol{\xi}_k = (x_{\zeta_{k,1}},\dots,x_{\zeta_{k,N_k}})$ for $k \in \{1,\dots,K\}$. Subsequently, $\operatorname{mdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}};\mathscr{P}) := \left(\sum_{\eta=1}^{N_1}\alpha_{\zeta_{1,\eta}},\dots,\sum_{\eta=1}^{N_K}\alpha_{\zeta_{K,\eta}}\right) \in \mathbb{N}^K$. For instance, the monomial $x_1^3x_2^5x_3^2$ has $\operatorname{mdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}};\mathscr{P}_1) = (8,2)$ for $\mathscr{P}_1 = ((x_1,x_2),x_3)$ and $\operatorname{mdeg}(\boldsymbol{x}^{\boldsymbol{\alpha}};\mathscr{P}_2) = (3,5,2)$ for $\mathscr{P}_2 = (x_1,x_2,x_3)$.

A polynomial $p = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}}\boldsymbol{x}^{\boldsymbol{\alpha}}$, with coefficients $c_{\boldsymbol{\alpha}}$ belonging to some algebraically closed field $\mathbb{F}$, is obtained by taking a finite linear combination of monomials over that field $\mathbb{F}$. The set of all polynomials forms a ring, denoted $\mathbb{F}[\boldsymbol{x}]$. The *monomial* support of a polynomial $\operatorname{supp}(p) := \{\boldsymbol{\alpha} \in \mathbb{N}^N : c_{\boldsymbol{\alpha}} \neq 0\}$ is the set of exponents for which the associated monomial coefficients are nonzero. A polynomial is called (multi-)homogeneous (with respect to some $\mathscr{P}$) if all monomial terms belonging to the support have the same (multi-)degree. The corresponding (multi-)degree of the monomials is also referred to as the (multi-)degree of the polynomial.

A root of a polynomial p is a point $\boldsymbol{x} \in \mathbb{C}^N$ such that $p(\boldsymbol{x}) = 0$. For a homogeneous polynomial, $\boldsymbol{x} = \boldsymbol{0}$ is always a trivial root. A nontrivial root $\boldsymbol{x} \neq \boldsymbol{0}$ of a homogeneous polynomial satisfies $p(\lambda\boldsymbol{x}) = 0$ for any $\lambda \in \mathbb{C}$. This invariance makes it natural to study the zero sets of homogeneous polynomials within projective geometry $\mathbb{P}^{N-1}(\mathbb{C})$, where $\boldsymbol{x}$ and $\lambda\boldsymbol{x}$ represent the same point. The $(N-1)$-dimensional complex projective space $\mathbb{P}^{N-1}(\mathbb{C})$ is defined as the set of all lines in $\mathbb{C}^N$ passing through the origin. Coordinates in this space are written using the colon notation $(x_1 : \ldots : x_N) \in \mathbb{P}^{N-1}(\mathbb{C})$. For multi-homogeneous polynomials, a *trivial* root is any point in which one of the disjoint variable groups $\boldsymbol{\xi}_k \in \mathbb{C}^{N_k}$ is zero for some $k \in \{1,\dots,K\}$. A nontrivial root $(\boldsymbol{\xi}_1,\dots,\boldsymbol{\xi}_K)$ is a root of p if and only if $(\lambda_1\boldsymbol{\xi}_1\dots,\lambda_K\boldsymbol{\xi}_K)$ is also a root for any $\lambda_k \in \mathbb{C}$. Consequently, the natural domain for studying the zero sets of multi-homogeneous polynomials is the multi-projective space given by the Cartesian product $\mathbb{P}^{N_1}(\mathbb{C}) \times \ldots \times \mathbb{P}^{N_K}(\mathbb{C})$.

2. The tensor-based Macaulay method.

A tensor-based method for computing all isolated common roots of homogeneous polynomial systems in projective space was introduced in recent works [49, 50, 53]. In this section, we generalize this approach to multi-homogeneous systems in multi-projective space, which naturally accommodates polynomial systems whose monomial support lies on a rectilinear grid. Owing to structural differences, polynomial systems with rectilinear support require a distinct treatment from the total-degree case. While much of the material in this section will be familiar to readers with a background in applied algebraic geometry, the following exposition aims to make the material accessible for a wider audience.

2.1. Polynomial systems with monomial support on a rectilinear grid.

Let $\{\phi_k\}_{k=0}^\infty$ denote a polynomial basis described by a three-term recurrence relation

$$\alpha_k\phi_{k+1}(x) = (x-\beta_k)\phi_k(x) - \gamma_k\phi_{k-1}(x), \quad \alpha_0\phi_1(x) = (x-\beta_0)\phi_0(x) \tag{2.1}$$

with $\phi_0(x) = 1$. The monomials obviously belong to this class of polynomial bases given the recurrence relation $\phi_{k+1}(x) = x\phi_k(x)$, but so do families of orthogonal polynomials on the real line[1] with respect to some positive weighting function $w : [a,b] \to \mathbb{R}_+$. Favard's theorem ensures that a polynomial basis $\{\phi_k\}_{k=0}^\infty$ satisfying the

[1]The monomials are, on the other hand, orthogonal on the unit circle.

orthogonality relations

$$\int_a^b \phi_k(x)\phi_l(x)w(x)\mathrm{d}x = 0, \qquad k \neq l,$$

can be represented using a recurrence such as in (2.1). One important class of orthogonal polynomials considered in this paper are the Chebyshev polynomials, which satisfy the recurrence $\frac{1}{2}\phi_{k+1}(x) = x\phi_k(x) - \frac{1}{2}\phi_{k-1}(x)$ with $\phi_1(x) = x\phi_0(x)$. These polynomials are orthogonal with respect to the weighting function $w(x) = (1-x^2)^{-1/2}$ on $[-1, 1]$.

Write $\phi_{\boldsymbol{k}}(\boldsymbol{x}) := \phi_{k_1}(x_1)\cdots\phi_{k_N}(x_N)$ and define the rectilinear grid

$$\mathscr{A}_N[\boldsymbol{d}] := \left\{\boldsymbol{k} \in \mathbb{N}^N : k_i \leq d_i,\, i = 1, \ldots, N\right\}.$$

We are interested in finding the common roots of the overdetermined polynomial system

$$\Sigma: \quad f^{(s)}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathscr{A}_N[\boldsymbol{d}^{(s)}]} c_{\boldsymbol{k}}^{(s)}\phi_{\boldsymbol{k}}(\boldsymbol{x}) = 0, \quad s \in \{1, \ldots, S\}, \quad S \geq N. \tag{2.2}$$

Herein, it is assumed that the number of roots R is finite. Generically, $f^{(s)} \in \Sigma$ has monomial support $\mathscr{A}_N[\boldsymbol{d}^{(s)}]$ since conversion of $f^{(s)}$ to monomial bases involve dense transformations. The coefficients of $f^{(s)}$ can be stored in a tensor of size $(d_1 + 1) \times \ldots \times (d_N + 1)$ and it is convenient to introduce the notation $\mathcal{C}^{(s)} = \mathrm{coeff}(g)$ to represent this tensor.

2.2. Structural differences with total-degree systems. Polynomial systems of the type (2.2) behave differently from total-degree systems whose support is confined to the N-dimensional simplex bounded by total degree d, i.e.,

$$\mathscr{B}_N[d] := \left\{\boldsymbol{k} \in \mathbb{N}^N : k_1 + k_2 + \ldots + k_N \leq d\right\}.$$

One notable difference is the growth in the number of polynomial coefficients. Systems of the type in (2.2) grow exponentially (unless additional structure is imposed) with $\prod_{n=1}^N d_n$ coefficients or d^N coefficients if $d_n = d$ for all $n \in \{1, \ldots, N\}$. On the other hand, total-degree systems with support $\mathscr{B}_N[d]$ grow combinatorially with $\binom{N+d}{N}$ coefficients. For low-degree systems, this difference can be stark: a multi-degree polynomial with $d_n = 1$ for $n \in \{1, \ldots, N\}$ has 2^N coefficients, whereas a total-degree two polynomial has only $(N + 1)(N + 2)/2$ coefficients.

Another significant difference is the number of solutions. For square systems $S = N$, Bernstein–Kushnirenko's theorem [23] states that the number of solutions equals the mixed volume generated by the convex hulls of the monomial support sets $\mathrm{supp}(f^{(s)})$, with $f^{(s)} \in \Sigma$. For systems of the form (2.2), this yields

$$\mathrm{perm}(\begin{bmatrix}\boldsymbol{d}^{(1)} & \cdots & \boldsymbol{d}^{(N)}\end{bmatrix}) := \sum_{\sigma\in\mathscr{S}_N}\prod_{n=1}^N d_i^{(\sigma(n))} \tag{2.3}$$

common roots in $(\mathbb{C} \setminus 0)^N$, where $\mathrm{perm}(\cdot)$ denotes the permanent and $\mathscr{S}_n$ the set of all permutations of $\{1, \ldots, N\}$. In the special case $\boldsymbol{d}^{(s)} = \boldsymbol{d}$ for all s, the system has $N!\prod_{n=1}^N d_n$ roots (or $N!d^N$ when all $d_n = d$), whereas a total-degree system has only $\prod_{s=1}^N d^{(s)}$ roots (or d^N when all $d_n = d$).

2.3. Homogenization. The roots of a nonhomogeneous system in affine space can be recovered from a homogeneous system via homogenization. In the standard formulation, the system Σ in (2.2) is transformed to

$$\Sigma_t : \quad g^{(s)}(t, \boldsymbol{x}) = t^{d_1^{(s)}+\cdots+d_N^{(s)}} f^{(s)}(\boldsymbol{x}/t) = 0, \quad s \in \{1, \ldots, S\}, \tag{2.4}$$

producing homogeneous polynomials of degree $d^{(s)} = d_1^{(s)} + \cdots + d_N^{(s)}$ through the introduction of a homogenization variable t. The roots of Σ and Σ_t are related as follows: $(t : x_1 : \cdots : x_N) \in \mathbb{P}^N(\mathbb{C})$ with $t \neq 0$ is a projective common root of Σ_t if and only if $\boldsymbol{x}/t \in \mathbb{C}^N$ is a common root of Σ in affine space. Consequently, solving Σ_t allows one to recover the solutions of the original system.

However, this approach can produce byproducts. If Σ_t has a projective root $(0 : x_1 : \cdots : x_N) \in \mathbb{P}^N(\mathbb{C})$ with $t = 0$, it does not correspond to any finite root of Σ. Instead, it represents a so-called root at infinity as the following example illustrates.

EXAMPLE 2.1. *Consider the pair of polynomial equations*

$$\Sigma : \qquad f^{(1)} := 1 - x - y + xy = 0, \quad f^{(2)} := 2 + x - y + 2xy = 0, \tag{2.5}$$

which has two common roots at $(\frac{1}{3}(1+\sqrt{5}), -1-\sqrt{5})$ *and* $(\frac{1}{3}(1-\sqrt{5}), -1+\sqrt{5})$. *Replacing* x *with* $\frac{x}{t}$ *and* y *with* $\frac{y}{t}$ *produces the homogenized equations*

$$\Sigma_t : \qquad g^{(1)} := t^2 - tx - ty + xy = 0, \quad g^{(2)} := 2t^2 + tx - ty + 2xy = 0. \tag{2.6}$$

The system (2.6) *has four projective roots. The projective roots* $(1 : \frac{1}{3}(1+\sqrt{5}) : -1-\sqrt{5})$ *and* $(1 : \frac{1}{3}(1-\sqrt{5}) : -1+\sqrt{5})$ *correspond with affine roots of the original system* (2.5). *However, the roots* $(0 : 1 : 0)$ *and* $(0 : 0 : 1)$ *are roots at infinity and do not relate to any roots of the original system* (2.5).

In applications, roots at infinity are not always of interest. When this is the case, there is a problem since under the homogenization (2.4), the roots at infinity correspond generically to the zeros of the system

$$\Sigma_\infty : \quad h_s(\boldsymbol{x}) = \boldsymbol{x}^{\boldsymbol{d}^{(s)}} = 0, \quad s \in \{1, \ldots, S\}.$$

Σ_∞ always possesses roots at infinity. For $N > 2$, the number of roots at infinity in Σ_∞ are even uncountable, regardless of the number of equations S or the number of affine roots R. Specifically, any $(x_1 : \cdots : x_N) \in \mathbb{P}^{N-1}(\mathbb{C})$ with $x_n = 0$ for some $n \in 1, \ldots, N$ is a common zero of Σ_∞. Fortunately, this complication is merely an artifact of an inappropriate homogenization strategy.

EXAMPLE 2.2. *Consider again the system* (2.5). *Instead of a single homogenization variable, we may introduce a homogenization variable for each coordinate by replacing* x *with* $\frac{x}{t}$ *and* y *with* $\frac{y}{v}$. *This produces the bihomogeneous system*

$$\Sigma_{(t,v)} : \qquad g^{(1)} := tv - xv - ty + xy = 0, \quad g^{(2)} := 2tv + xv - ty + 2xy = 0, \tag{2.7}$$

with $(t : x, v : y) \in \mathbb{P}^1(\mathbb{C}) \times \mathbb{P}^1(\mathbb{C})$. *Unlike* (2.6), *the system* (2.7) *has only the roots* $(1 : \frac{1}{3}(1+\sqrt{5}), 1 : -1-\sqrt{5})$ *and* $(1 : \frac{1}{3}(1-\sqrt{5}), 1 : -1+\sqrt{5})$ *and no roots at infinity.*

The implications of Example 2.2 are not coincidental. The homogenization of Σ to

$$\Sigma_{\boldsymbol{t}} : \quad g^{(s)}(\boldsymbol{t}, \boldsymbol{x}) = \boldsymbol{t}^{\boldsymbol{d}^{(s)}} f^{(s)}(x_1/t_1, \ldots, x_N/t_N) = 0, \quad s \in \{1, \ldots, S\}, \tag{2.8}$$

produces a generic system of multi-homogeneous polynomials with $\mathrm{mdeg}(g^{(s)};\mathscr{P}) = \boldsymbol{d}^{(s)}$ and $\mathscr{P} = ((t_n, x_n))_{n=1}^{N}$. The polynomials in $\Sigma_{\boldsymbol{t}}$ have the structure

$$\Sigma_{\boldsymbol{t}}: \quad g^{(s)}(\boldsymbol{t},\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathscr{A}_N[\boldsymbol{d}^{(s)}]} c_{\boldsymbol{k}}^{(s)} \hat{\phi}_{\boldsymbol{d}^{(s)},\boldsymbol{k}}(\boldsymbol{t},\boldsymbol{x}) = 0,$$

where $\hat{\phi}_{\boldsymbol{d},\boldsymbol{k}}(\boldsymbol{t},\boldsymbol{x}) = \hat{\phi}_{d_1,k_1}(t_1,x_1)\cdots\hat{\phi}_{d_N,k_N}(t_N,x_N)$ and

$$\hat{\phi}_{d,k}(t,x) := t^d \phi_k(x/t). \tag{2.9}$$

For square systems, the multi-homogeneous Bézout theorem [40] guarantees that the number of multi-projective roots of (2.8) equals the coefficient of $t_1\cdots t_N$ in the polynomial $\prod_{n=1}^{N}(d_1^{(n)}t_1 + \cdots + d_N^{(n)}t_N)$. One can verify that this number matches the root count in (2.3). Thus, the homogenization in (2.8) generically does not produce roots at infinity for the system in (2.2).

2.4. The (multi-homogeneous) Macaulay matrix. To solve $\Sigma_{\boldsymbol{t}}$ in (2.8), we construct a multi-homogeneous variant of the Macaulay matrix. In contrast to the Macaulay matrices in [49, 50, 53], which describe maps between homogeneous polynomials, this variant acts on multi-homogeneous polynomials. Before introducing a coordinate-free construction of the Macaulay matrices using resultant maps [47, Definition 2.4], we first illustrate the essential differences between the two variants with the aid of Examples 2.1 and 2.2.

EXAMPLE 2.3. *For the homogeneous system Σ_t in (2.6), the Macaulay matrix $\mathbf{M}(D;\Sigma_t)$ at degree D is the matrix whose rows span the polynomials*

$$\left\{ t^{D-2-i-j}x^i y^j \cdot g^{(s)} : \quad (i,j)\in\mathscr{B}_2[D-d^{(s)}],\ s\in\{1,2\} \right\}.$$

At $D=3$, this matrix takes on the shape

$$\mathbf{M}(3;\Sigma_t) = \begin{array}{c} \\ t\cdot g_1 \\ t\cdot g_2 \\ x\cdot g_1 \\ x\cdot g_2 \\ \hline y\cdot g_1 \\ y\cdot g_2 \end{array} \left[\begin{array}{cccc|ccc|ccc|c} t^3 & t^2x & tx^2 & x^3 & t^2y & txy & x^2y & ty^2 & xy^2 & y^3 \\ \hline 1 & -1 & & & -1 & 1 & & & & \\ 2 & 1 & & & -1 & 2 & & & & \\ & 1 & -1 & & & -1 & 1 & & & \\ & 2 & 1 & & & -1 & 2 & & & \\ \hline & & & & 1 & -1 & & -1 & 1 & \\ & & & & 2 & 1 & & -1 & 2 & \end{array}\right].$$

For the bihomogeneous system $\Sigma_{(t,v)}$, the rows of $\mathbf{M}((D_x,D_y);\Sigma_{\boldsymbol{t}})$ at bidegree (D_x,D_y) spans the polynomials

$$\left\{ t^{D_x-1-i}x^i v^{D_y-1-j}y^j \cdot g^{(s)} : \quad (i,j)\in\mathscr{A}_2[D-d^{(s)}],\ s\in\{1,2\} \right\}.$$

At $(D_x,D_y)=(2,2)$, this matrix takes on the shape

$$\mathbf{M}((2,2);\Sigma_{(t,v)}) = \begin{array}{c} \\ tv\cdot g_1 \\ tv\cdot g_2 \\ xv\cdot g_1 \\ xv\cdot g_2 \\ \hline ty\cdot g_1 \\ ty\cdot g_2 \\ xy\cdot g_1 \\ xy\cdot g_2 \end{array} \left[\begin{array}{ccc|ccc|ccc} t^2v^2 & txv^2 & x^2v^2 & t^2vy & txvy & x^2vy & t^2y^2 & txy^2 & x^2y^2 \\ \hline 1 & -1 & & -1 & 1 & & & & \\ 2 & 1 & & -1 & 2 & & & & \\ & 1 & -1 & & -1 & 1 & & & \\ & 2 & 1 & & -1 & 2 & & & \\ \hline & & & 1 & -1 & & -1 & 1 & \\ & & & 2 & 1 & & -1 & 2 & \\ & & & & 1 & -1 & & -1 & 1 \\ & & & & 2 & 1 & & -1 & 2 \end{array}\right].$$

The matrices in Example 2.3 arise from the following construction. Let $\mathbb{C}_d[t, \boldsymbol{x}]$ denote the vector space of all homogeneous polynomials of degree d. Likewise, let $\mathbb{C}_{\boldsymbol{d}}[\boldsymbol{t}, \boldsymbol{x}]$ be the vector space of all multi-homogeneous polynomials of multi-degree $\boldsymbol{d}$ for the partition $\mathscr{P} = ((t_1, x_1), \ldots, (t_N, x_N))$. For Σ_t in (2.4), we define the resultant map $\mathscr{M}(D; \Sigma_t) : \mathbb{C}_{D-d^{(1)}}[t, \boldsymbol{x}] \times \ldots \times \mathbb{C}_{D-d^{(S)}}[t, \boldsymbol{x}] \to \mathbb{C}_D[t, \boldsymbol{x}]$ by the linear operator

$$\mathscr{M}(D; \Sigma_t) : \quad (h^{(1)}, \ldots, h^{(S)}) \mapsto \sum_{s=1}^{S} h^{(s)} g^{(s)}. \tag{2.10}$$

For $\Sigma_{\boldsymbol{t}}$ in (2.8), the resultant map takes the same functional form,

$$\mathscr{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}}) : \quad (h^{(1)}, \ldots, h^{(S)}) \mapsto \sum_{s=1}^{S} h^{(s)} g^{(s)}, \tag{2.11}$$

but acts as $\mathscr{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}}) : \mathbb{C}_{\boldsymbol{D}-\boldsymbol{d}^{(1)}}[\boldsymbol{t}, \boldsymbol{x}] \times \ldots \times \mathbb{C}_{\boldsymbol{D}-\boldsymbol{d}^{(S)}}[\boldsymbol{t}, \boldsymbol{x}] \to \mathbb{C}_{\boldsymbol{D}}[\boldsymbol{t}, \boldsymbol{x}]$. The Macaulay matrices $\mathbf{M}(D; \Sigma_t)$ and $\mathbf{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}})$ at degree D and multi-degree $\boldsymbol{D}$ are defined as the transposes of the operators (2.10) and (2.11) in some basis, respectively.

EXAMPLE 2.4. *For the homogeneous system Σ_t from* (2.6)*, the domain and codomain of the operator in* (2.10) *can be written as*

$$h^{(s)} = \sum_{\boldsymbol{j} \in \mathscr{B}_2[D-2]} a_{\boldsymbol{j}}^{(s)} t^{D-2-j_1-j_2} x^{j_1} y^{j_2}, \quad \sum_{s=1}^{S} h^{(s)} g^{(s)} = \sum_{\boldsymbol{j} \in \mathscr{B}_2[D]} b_{\boldsymbol{j}} t^{D-j_1-j_2} x^{j_1} y^{j_2}$$

so that $\mathscr{M}(D; \Sigma_t) : (h^{(1)}, h^{(2)}) \longmapsto h^{(1)}(t^2 - tx - ty + xy) + h^{(2)}(2t^2 + tx - ty + 2xy)$ defines a map between spaces of homogeneous polynomials. In contrast, for the bihomogeneous system $\Sigma_{(t,v)}$ in (2.6)*, the domain and codomain of operator* (2.11) *are*

$$h^{(s)} = \sum_{\boldsymbol{j} \in \mathscr{A}_2[\boldsymbol{D}-\boldsymbol{1}]} a_{\boldsymbol{j}}^{(s)} t^{D_1-1-j_1} v^{D_2-1-j_2} x^{j_1} y^{j_2}, \quad \sum_{s=1}^{S} h^{(s)} g^{(s)} = \sum_{\boldsymbol{j} \in \mathscr{A}_2[\boldsymbol{D}]} b_{\boldsymbol{j}} t^{D_1-j_1} v^{D_2-j_2} x^{j_1} y^{j_2},$$

and the corresponding operator $\mathscr{M}(\boldsymbol{D}; \Sigma_{(t,v)}) : (h^{(1)}, h^{(2)}) \longmapsto h^{(1)}(tv - xv - ty + xy) + h^{(2)}(2tv + xv - ty + 2xy)$ acts between spaces of bihomogeneous polynomials. Thus, if $\imath(s, \boldsymbol{i})$ and $\jmath(\boldsymbol{j})$ denote maps that convert multi-indices to linear indices, $[\mathbf{M}(D; \Sigma_t)]_{\imath(s,\boldsymbol{i}),\jmath(\boldsymbol{j})}$ and $\left[\mathbf{M}(D_x, D_y; \Sigma_{(t,v)})\right]_{\imath(s,\boldsymbol{i}),\jmath(\boldsymbol{j})}$ represent the contributions of coefficient $a_{\boldsymbol{i}}^{(s)}$ to coefficient $b_{\boldsymbol{j}}$. For the selected bases, one attains the multi-level (quasi-)Toeplitz matrices from Example 2.3.

The structure of the Macaulay matrices changes when using a different basis, but fundamentally they always encode the operation of forming polynomial combinations. In section 3, we provide explicit constructions of multi-homogeneous Macaulay matrices in the monomial and Chebyshev basis. For now, note that $\mathbf{M}(D; \Sigma_t)$ has $\sum_{s=1}^{S} \binom{N+D-d^{(s)}}{N}$ rows and $\binom{N+D}{N}$ columns, whereas $\mathbf{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}})$ is larger and has $\sum_{s=1}^{S} \prod_{n=1}^{N} (D_n - d_n^{(s)} + 1)$ rows and $\prod_{n=1}^{N} (D_n + 1)$ columns.

2.5. The Macaulay null space and system roots. As the following example suggests, the null spaces of the Macaulay matrices $\mathbf{M}(D; \Sigma_t)$ and $\mathbf{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}})$ are intimately tied to the (multi-)projective roots of Σ_t and $\Sigma_{\boldsymbol{t}}$, respectively.

EXAMPLE 2.5. *The nullity of* $\mathbf{M}(3;\Sigma_t)$ *in Theorem* 2.4 *is four. The Vandermonde vectors*

$$\boldsymbol{v}(t,x,y) := \begin{bmatrix} t^3 & t^2x & tx^2 & x^3 & t^2y & txy & x^2y & ty^2 & xy^2 & y^3 \end{bmatrix}^\top$$

evaluated at the roots $(1 : \frac{1}{3}(1+\sqrt{5}) : -1-\sqrt{5})$, $(1 : \frac{1}{3}(1-\sqrt{5}) : -1+\sqrt{5})$, $(0:1:0)$, *and* $(0:0:1)$ *belong to the null space of* $\mathbf{M}(3;\Sigma_t)$. *In fact, the vectors form a basis for the null space. The nullity of* $\mathbf{M}((2,2);\Sigma_{(t,v)})$ *in Example 2.4 is two. The Vandermonde vectors*

$$\boldsymbol{v}(t,x,v,y) := \begin{bmatrix} t^2v^2 & txv^2 & x^2v^2 & t^2vy & txvy & x^2vy & t^2y^2 & txy^2 & x^2y^2 \end{bmatrix}^\top$$

evaluated at the roots $(1 : \frac{1}{3}(1+\sqrt{5}), 1 : -1-\sqrt{5})$ *and* $(1 : \frac{1}{3}(1-\sqrt{5}), 1 : -1+\sqrt{5})$ *form a null space basis for* $\mathbf{M}(2,2;\Sigma_{(t,v)})$.

The observation that Vandermonde vectors evaluated at a system's roots belong to the null space is independent of the choice of basis in which the Macaulay matrix is expressed. This follows directly from the fact that the ranges of the operators $\mathscr{M}(D;\Sigma_t)$ and $\mathscr{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$ lie within the ideals generated by their respective systems, and all polynomials in an ideal must vanish on the corresponding variety.

For the system $\Sigma_{\boldsymbol{t}}$ in (2.8), this implies the following. Let $\boldsymbol{v}_D(\tau,\chi) \in \mathbb{C}^{D+1}$ denote the Vandermonde vector with entries

$$[\boldsymbol{v}_D(\tau,\chi)]_j = \hat{\phi}_{D,j}(\tau,\chi) \tag{2.12}$$

for $j \in \{0,1,\ldots,D\}$ and define

$$\operatorname{vec}(\mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau},\boldsymbol{\chi})) := \boldsymbol{v}_{D_1}(\tau_1,\chi_1) \otimes \cdots \otimes \boldsymbol{v}_{D_N}(\tau_N,\chi_N), \tag{2.13}$$

as the multivariate Vandermonde vector obtained by taking Kronecker products of univariate Vandermonde vectors. If $(\tau_1 : \chi_1, \ldots, \tau_N : \chi_N) \in \mathbb{P}^1(\mathbb{C}) \times \ldots \times \mathbb{P}^1(\mathbb{C})$ is a root of $\Sigma_{\boldsymbol{t}}$, then

$$\operatorname{vec}(\mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau},\boldsymbol{\chi})) \in \operatorname{null} \mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}}).$$

A similar property can be established for $\mathbf{M}(D;\Sigma_t)$. However, unlike the construction in (2.13), the Vandermonde vectors of $\mathbf{M}(D;\Sigma_t)$ cannot readily be obtained by vectorizing a rank-one tensor. Whereas (2.13) directly reshapes into a rank-one tensor, the Vandermonde vectors for $\mathbf{M}(D;\Sigma_t)$ require more complicated tensor embeddings.

2.6. The Hilbert–Serre theorem and its implications. At sufficiently high (multi-)degrees, the nullities of the Macaulay matrices stabilize to a constant equal to the number of roots. This stabilization follows directly from the classical Hilbert–Serre theorem in commutative algebra and algebraic geometry [21]. The dimensions of the cokernels of $\mathscr{M}(D;\Sigma_t)$ and $\mathscr{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$, viewed as functions of their (multi-)degrees, coincide with the Hilbert function $H_f(D)$ and its multi-graded analogue $H_f(\boldsymbol{D})$. Hilbert–Serre asserts that for sufficiently large degrees D^* and multi-degrees $\boldsymbol{D}^*$, the functions $H_f(D)$ and $H_f(\boldsymbol{D})$ eventually agree with a polynomial; see [26] and [32] for discussions of the graded and multigraded cases, respectively. When the (multi)homogeneous systems have only isolated (multi)projective roots, this polynomial becomes constant, and its value is precisely the number of roots counted with multiplicity[2].

[2]In the general case, the leading term of the Hilbert polynomial encodes valuable geometric information about the underlying variety. For the standard graded case, e and r in the leading term $\frac{e}{(r-1)!}d^{r-1}$ are, respectively, the dimension and degree of the variety. The latter is defined as the number of intersection points of the variety with a general linear subspace of co-dimension r. When $r = 1$, this degree coincides with the number of isolated roots [21, Section 1.9].

Since the cokernels of $\mathscr{M}(D;\Sigma_t)$ and $\mathscr{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$ are isomorphic to the null spaces of their corresponding Macaulay matrices, the following key result follows directly from the discussion in subsection 2.5.

PROPOSITION 2.6. *Suppose that* $\Sigma_{\boldsymbol{t}}$ *in* (2.8) *has a finite set of* R distinct[3] *roots* $\{(\tau_1^{(r)} : \chi_1^{(r)}, \ldots, \tau_N^{(r)} : \chi_N^{(r)})\}_{r=1}^R \subset \mathbb{P}^1(\mathbb{C}) \times \ldots \times \mathbb{P}^1(\mathbb{C})$ *. Then, for a sufficiently large* $\boldsymbol{D} \geq \boldsymbol{D}^*$ *after which the Hilbert function is known to stablize to the constant* R*, the collection of Vandermonde vectors* $\{\mathrm{vec}(\mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau}^{(r)}, \boldsymbol{\chi}^{(r)}))\}_{r=1}^R \subset \mathbb{C}^{\prod_{n=1}^N (D_n+1)}$ *forms a basis for* $\operatorname{null} \mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$.

A similar statement holds for Σ_t in (2.4) with respect to the null space of $\mathbf{M}(D;\Sigma_{\boldsymbol{t}})$. The degree at which the Hilbert functions stabilize varies per system. For square homogeneous systems, an upper bound for this stabilization degree is well known and given by $D^* \leq \sum_{s=1}^N d^{(s)} - N$ [9, Corollary 4.3]. An upper bound for the square multi-homogeneous case has also been derived [9, Proposition 4.3]. For $\Sigma_{\boldsymbol{t}}$, this bound reduces to $D_n^* \leq \sum_{s=1}^N \boldsymbol{d}_n^{(s)} - 1$ for $n \in \{1, \ldots, N\}$.

2.7. Reduction to a tensor decomposition problem. Proposition 2.6 forms the foundation of the tensor-based method for finding isolated roots of the polynomial system $\Sigma_{\boldsymbol{t}}$. The central idea is to recover the Vandermonde basis from any computed null space basis of the Macaulay matrix. One approach to achieve this is by exploiting the shift property of the Vandermonde vectors in (2.12). This produces a tensor decomposition problem. Define

$$\mathbf{S}_D := \begin{bmatrix} \beta_0 & \alpha_0 & 0 & \cdots & \cdots & 0 \\ \gamma_1 & \beta_1 & \alpha_1 & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \gamma_D & \beta_D & \alpha_D \end{bmatrix} \in \mathbb{R}^{D \times (D+1)}. \tag{2.14}$$

where $\{(\alpha_d, \beta_d, \gamma_d)\}_{d=0}^D$ originate from the recurrence relation (2.1). By combining (2.9) with (2.1), the product $\mathbf{S}_D \boldsymbol{v}_D(\tau, \chi)$ reduces to

$$\begin{bmatrix} \tau^d \left(\beta_0 \phi_0\left(\frac{\chi}{\tau}\right) + \alpha_0 \phi_1\left(\frac{\chi}{\tau}\right)\right) \\ \vdots \\ \tau^d \left(\gamma_D \phi_{D-2}\left(\frac{\chi}{\tau}\right) + \beta_D \phi_{D-1}\left(\frac{\chi}{\tau}\right) + \alpha_D \phi_D\left(\frac{\chi}{\tau}\right)\right) \end{bmatrix} = \begin{bmatrix} \tau^d \cdot \frac{\chi}{\tau} \phi_0(\frac{\chi}{\tau}) \\ \vdots \\ \tau^d \cdot \frac{\chi}{\tau} \phi_{D-1}\left(\frac{\chi}{\tau}\right) \end{bmatrix} = \chi \boldsymbol{v}_{D-1}(\tau, \chi),$$

making the shift structure explicit. Similarly, one can show that $\mathbf{I}_{D,D+1} \boldsymbol{v}_D(\tau, \chi) = \tau \boldsymbol{v}_{D-1}(\tau, \chi)$. Subsequently, these properties produce the following collection of identities for (2.13):

$$\begin{aligned} \mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau}, \boldsymbol{\chi}) \ \cdot_1 \mathbf{I}_{D_1, D_1+1} \ \cdots \ \cdot_N \mathbf{I}_{D_N, D_N+1} &= \left(\textstyle\prod_{n=1}^N \tau_n\right) \mathcal{V}_{\boldsymbol{D}-1}(\boldsymbol{\tau}, \boldsymbol{\chi}), \\ \mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau}, \boldsymbol{\chi}) \ \cdot_1 \mathbf{S}_{D_1} \ \cdots \ \cdot_N \mathbf{I}_{D_N, D_N+1} &= \left(\chi_1 \textstyle\prod_{n=2}^N \tau_n\right) \mathcal{V}_{\boldsymbol{D}-1}(\boldsymbol{\tau}, \boldsymbol{\chi}), \\ &\vdots \\ \mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau}, \boldsymbol{\chi}) \ \cdot_1 \mathbf{I}_{D_1, D_1+1} \ \cdots \ \cdot_N \mathbf{S}_{D_N} &= \left(\chi_n \textstyle\prod_{n=1}^{N-1} \tau_n\right) \mathcal{V}_{\boldsymbol{D}-1}(\boldsymbol{\tau}, \boldsymbol{\chi}). \end{aligned} \tag{2.15}$$

[3] A generalization for systems with repeated roots is possible, but requires the introduction of generalized Vandermonde bases, see [50].

Let $\boldsymbol{D} \geq \boldsymbol{D}^*$ and suppose that $\{\mathrm{vec}(\mathcal{N}_r)\}_{r=1}^{\tilde{R}} \subset \mathbb{C}^{\prod_{n=1}^{N}(D_n+1)}$ forms a collection of $\tilde{R}$ linearly independent null space vectors of $\mathbf{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}})$. A tensor $\mathcal{N}$ of shape $(D_1+1) \times \cdots \times (D_N+1) \times \tilde{R}$ can be constructed by concatenating $\{\mathcal{N}_r\}_{r=1}^{\tilde{R}}$ along the $(N+1)$-th mode, i.e., $\mathcal{N} = \mathrm{cat}(\{\mathcal{N}_r\}_{r=1}^{\tilde{R}}, N+1)$. Subsequently, we form the tensor $\mathcal{T} \in \mathbb{C}^{(N+1)\times D_1 \times \ldots \times D_N \times \tilde{R}}$, where

$$\begin{aligned}
\mathcal{T}(1,:,\ldots,:,:) &= \mathcal{N} \cdot_1 \mathbf{I}_{D_1,D_1+1} \cdots \cdot_N \mathbf{I}_{D_N,D_N+1}, \\
\mathcal{T}(2,:,\ldots,:,:) &= \mathcal{N} \cdot_1 \mathbf{S}_{D_1} \cdots \cdot_N \mathbf{I}_{D_N,D_N+1}, \\
&\vdots \\
\mathcal{T}(N+1,:,\ldots,:,:) &= \mathcal{N} \cdot_1 \mathbf{I}_{D_1,D_1+1} \cdots \cdot_N \mathbf{S}_{D_N}.
\end{aligned}$$

Since Proposition 2.6 ensures the existence of a "mixing" matrix $\mathbf{W} \in \mathbb{C}^{\tilde{R}\times R}$ such that $\mathcal{N} = \mathcal{V}_{\boldsymbol{D}} \cdot_{N+1} \mathbf{W}$, where $\mathcal{V}_{\boldsymbol{D}} := \mathrm{cat}(\{\mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau}^{(r)}, \boldsymbol{\chi}^{(r)})\}_{r=1}^{R}, N+1)$. By applying the identities (2.15), the tensor $\mathcal{T}$ can be shown to admit the CPD

$$\mathcal{T} = [\![\mathbf{X}, \mathbf{V}_{D_1-1,1}, \ldots, \mathbf{V}_{D_N-1,N}, \mathbf{W}]\!], \tag{2.16}$$

where $\mathbf{V}_{D_n,n} := \begin{bmatrix} \boldsymbol{v}_{D_n}(\tau_n^{(1)}, \chi_n^{(1)}) & \cdots & \boldsymbol{v}_{D_n}(\tau_n^{(R)}, \chi_n^{(R)}) \end{bmatrix}$ for $n \in \{1, \ldots, N\}$ and

$$\mathbf{X} = \begin{bmatrix} \prod_{n=1}^{N} \tau_n^{(1)} & \prod_{n=1}^{N} \tau_n^{(2)} & \cdots & \prod_{n=1}^{N} \tau_n^{(R)} \\ \chi_1^{(1)} \prod_{n=2}^{N} \tau_n^{(1)} & \chi_1^{(2)} \prod_{n=2}^{N} \tau_n^{(2)} & \cdots & \chi_1^{(R)} \prod_{n=2}^{N} \tau_n^{(R)} \\ \vdots & & & \\ \chi_N^{(1)} \prod_{n=1}^{N-1} \tau_n^{(1)} & \chi_N^{(2)} \prod_{n=1}^{N-1} \tau_n^{(2)} & \cdots & \chi_1^{(R)} \prod_{n=1}^{N-1} \tau_n^{(R)} \end{bmatrix}.$$

Note that the columns of $\mathbf{X}$ are the multi-homogeneous roots of the system $\Sigma_{\boldsymbol{t}}$. Specifically, if $\tau_n^{(r)} \neq 0$ for all $n \in \{1, \ldots, N\}$, the r-th column of $\mathbf{X}$ corresponds with an affine root of the original system Σ as it can be divided by $\prod_{n=1}^{N} \tau_n^{(r)}$ to produce $(\chi_1^{(r)}/\tau_1^{(r)}, \ldots, \chi_N^{(r)}/\tau_N^{(r)})$ in the second-to-last entry. On the other hand, $\{\mathbf{V}_{D_n-1,n}\}_{n=1}^{N}$ holds the corresponding Vandermonde vectors.

Provided that $\mathbf{W}$ has no proportional columns, essential uniqueness of the CPD (2.16) is guaranteed if the roots of $\Sigma_{\boldsymbol{t}}$ are distinct. If $\tilde{R} = 1$, $\mathbf{W}$ must contain non-zero entries and the roots of $\Sigma_{\boldsymbol{t}}$ must be distinct in each coordinate[4]. This observation makes (2.16) more versatile than it may appear initially. Owing to the uniqueness properties of higher-order tensor decompositions [19], essential uniqueness generically holds even for $\tilde{R} = 1$, that is, when only a single null space vector is available. For the Krylov methods in section 4, this property can lead to significant computational savings, since a full null space basis is no longer required to find all roots.

Nonetheless, for the special case $\tilde{R} = R$, (2.16) is analogous to computing the eigendecompositions of the multiplication tables of the quotient rings generated by the underlying ideals, after an appropriate reshaping of $\mathcal{T}$ into a third-order tensor; see, e.g., [50]. Since these multiplication tables commute, they can be concatenated into a higher-order tensor to enable simultaneous diagonalization. However, the decomposition (2.16) exploits more than mere commutativity: it also leverages the separability of $\mathcal{V}_{\boldsymbol{D}}(\boldsymbol{\tau}, \boldsymbol{\chi})$ as described in (2.13).

[4]If $\Sigma_{\boldsymbol{t}}$ has roots with multiplicities greater than one, one must instead resort to block term decompositions; see, e.g., [50].

3. Fast matrix-vector products for Macaulay matrices. Section 2 introduced the multi-homogeneous Macaulay matrix as the transpose representation of the operator (2.11) in a chosen basis. In this section, we examine its structure in the monomial and Chebyshev bases and show how these structures enable efficient matrix–vector products. Such fast matrix–vector operations are essential for the performance of the Krylov-based null space methods presented in section 4.

3.1. Efficient matrix-vector multiplication in the monomial basis. Let Σ in (2.2) be expressed in the monomial basis $\phi_k(x) = x^k$. After homogenization, the basis terms become $\hat{\phi}_{d,k}(t,x) = t^{d-k}x^k$. Representing the domain of the operator (2.11) in the same basis, the entries of $\mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$ take the form

$$[\mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})]_{\imath(s,\boldsymbol{i}),\jmath(\boldsymbol{j})} = \begin{cases} c^{(s)}_{\boldsymbol{j}-\boldsymbol{i}} & \text{if } \boldsymbol{j}-\boldsymbol{i} \in \mathscr{A}_N[\boldsymbol{d}^{(s)}], \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

In the monomial basis, $\mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$ is a vertical concatenation of S *multi-level* Toeplitz matrices $\mathbf{M}(\boldsymbol{D};g^{(s)}) \in \mathbb{C}^{\prod_{n=1}^{N}(D_n-d_n^{(s)}+1)\times\prod_{n=1}^{N}(D_n+1)}$, associated with the individual polynomial equations.

The Toeplitz structures can be leveraged to construct efficient matrix-vector multiplication algorithms. In the one-dimensional case with $g = \sum_{k=0}^{d} c_k t^{d-k}x^k$, the Macaulay matrices reduces to

$$\mathbf{M}(D;g) = \begin{bmatrix} c_0 & c_1 & \cdots & c_d & & \\ & c_0 & c_1 & \cdots & c_d & \\ & & \ddots & \ddots & & \ddots \\ & & & c_0 & c_1 & \cdots & c_d \end{bmatrix} \in \mathbb{C}^{(D-d+1)\times(D+1)}.$$

The matrix $\mathbf{M}(D;g)$ can be embedded into the top $D-d+1$ rows of the circulant matrix

$$\mathrm{Circ}(\boldsymbol{a}) = \begin{bmatrix} a_0 & a_1 & \cdots & a_{D-1} & a_D \\ a_D & a_0 & a_1 & & a_{D-1} \\ \vdots & a_D & a_0 & \ddots & \vdots \\ a_2 & & \ddots & \ddots & a_1 \\ a_1 & a_2 & \cdots & a_D & a_0 \end{bmatrix} \in \mathbb{C}^{(D+1)\times(D+1)}$$

by setting $\boldsymbol{a} = (\boldsymbol{c}, \mathbf{0}_{D-d}) \in \mathbb{C}^{D+1}$. Using the convolution theorem

$$\mathrm{Circ}(\boldsymbol{a}) = \mathbf{F}^*_{D+1}\,\mathrm{diag}(\sqrt{D+1}\mathbf{F}^*_{D+1}\boldsymbol{a})\mathbf{F}_{D+1},$$

where $[\mathbf{F}_{D+1}]_{kl} = \frac{1}{\sqrt{D+1}}e^{-i\frac{2\pi}{D+1}(k-1)(l-1)}$ is the standard discrete Fourier transform (DFT) matrix in $D+1$ points, a fast matrix-vector product is possible by utilizing fast Fourier transforms (FFT).

The same Toeplitz-to-circulant embedding strategy generalizes naturally to higher dimensions. By setting $\mathcal{A} \in \mathbb{C}^{(D_1+1)\times\ldots\times(D_N+1)}$ equal to $\mathcal{C} \in \mathbb{C}^{(d_1+1)\times\ldots\times(d_N+1)}$ at $\mathcal{A}_{\mathbf{1}:(\boldsymbol{d}+1)}$ and zero elsewhere, $\mathbf{M}(\boldsymbol{D};\boldsymbol{g})$ can be embedded into the multi-level circulant matrix $\mathrm{Circ}(\mathcal{A}) \in \mathbb{C}^{\prod_{n=1}^{N}(D_n+1)\times\prod_{n=1}^{N}(D_n+1)}$ with entries[5]

$$[\mathrm{Circ}(\mathcal{A})]_{\jmath(\boldsymbol{i})\jmath(\boldsymbol{j})} = a_{\boldsymbol{i}-\boldsymbol{j} \bmod(\boldsymbol{D}+1)}.$$

[5]Here, $\boldsymbol{i}-\boldsymbol{j} \mod (\boldsymbol{D}+1) = (j_1 - i_1 \mod (D_1+1), \ldots, j_N - i_N \mod(D_N+1))$ describes an element-wise application of the mod operation.

Given the decomposition

$$\operatorname{Circ}(\mathcal{A}) = \left(\bigotimes_{n=1}^{N} \mathbf{F}^*_{D_n+1}\right) \operatorname{diag}\left(\left(\bigotimes_{n=1}^{N} \tilde{\mathbf{F}}^*_{D_n+1}\right) \operatorname{vec}(\mathcal{A})\right) \left(\bigotimes_{n=1}^{N} \mathbf{F}_{D_n+1}\right), \tag{3.2}$$

where $\tilde{\mathbf{F}}_{D_n+1} = \sqrt{D_n+1}\mathbf{F}_{D_n+1}$, the products $\mathbf{M}(\boldsymbol{D};\boldsymbol{g})\operatorname{vec}(\mathcal{X})$ and $\operatorname{vec}(\mathcal{Y})^*\mathbf{M}(\boldsymbol{D};\boldsymbol{g})$ can be computed efficiently using FFTs. Algorithm 3.1 outlines the procedure, where MV and MVA denote the matrix–vector and adjoint matrix–vector operations, respectively. The functions $\mathrm{FFT}_n(\mathcal{A}, P)$ and $\mathrm{IFFT}_n(\mathcal{A}, P)$ refer to the P-point FFT and inverse FFT algorithms applied along the mode n. Since the Krylov methods in section 4 require repeated matrix–vector products, the gray lines in algorithm 3.1 indicate computations that can be cached. Table 1 shows that Algorithm 3.1 significantly reduces memory usage and flop count compared with naive multiplication, lowering the time complexity from quadratic to log-linear in the vector size. Nevertheless, the total cost still grows exponentially with the number of variables, as the sizes of the vectors $\operatorname{vec}(\mathcal{X})$ and $\{\operatorname{vec}(\mathcal{Y}^{(s)})\}_{s=1}^{S}$ themselves scale exponentially.

Algorithm 3.1 Right and left multiplication in the monomial basis

function MV($\Sigma_{\boldsymbol{t}}$, $\boldsymbol{D}$, $\mathcal{X}$)
  for $n \in \{1,\dots,N\}$ **do**
    $\mathcal{X} \leftarrow \mathrm{FFT}_n(\mathcal{X}, D_n+1)$
  for $s \in \{1,\dots,S\}$ **do**
    for $n \in \{1,\dots,N\}$ **do**
      $\mathcal{C}^{(s)} \leftarrow \mathrm{IFFT}_n(\mathcal{C}^{(s)}, D_n+1)$
      $\mathcal{C}^{(s)} \leftarrow (D_n+1)^{\frac{1}{2}}\mathcal{C}^{(s)}$
    $\mathcal{Y}^{(s)} \leftarrow \mathcal{X} * \mathcal{C}^{(s)}$
    for $n \in \{1,\dots,N\}$ **do**
      $\mathcal{Y}^{(s)} \leftarrow \mathrm{IFFT}_n(\mathcal{Y}^{(s)}, D_n+1)$
    $\mathcal{Y}^{(s)} \leftarrow \left[\mathcal{Y}^{(s)}\right]_{\mathbf{1}:(\boldsymbol{D}-\boldsymbol{d}^{(s)}+\mathbf{1})}$
  return $\left\{\mathcal{Y}^{(s)}\right\}_{s=1}^{S}$

function MVA($\Sigma_{\boldsymbol{t}}$, $\boldsymbol{D}$, $\{\mathcal{Y}^{(s)}\}_{s=1}^{S}$)
  Init zero $\mathcal{X} \in \mathbb{C}^{(D_1+1)\times\dots\times(D_N+1)}$.
  for $s \in \{1,\dots,S\}$ **do**
    for $n \in \{1,\dots,N\}$ **do**
      $\mathcal{C}^{(s)} \leftarrow \mathrm{IFFT}_n(\mathcal{C}^{(s)}, D_n+1)$
      $\mathcal{C}^{(s)} \leftarrow (D_n+1)^{\frac{1}{2}}\mathcal{C}^{(s)}$
      $\mathcal{Y}^{(s)} \leftarrow \mathrm{FFT}_n(\mathcal{Y}^{(s)}, D_n+1)$
    $\mathcal{X} \leftarrow \mathcal{X} + \mathcal{Y}^{(s)} * \mathcal{C}^{(s)}$
  for $n \in \{1,\dots,N\}$ **do**
    $\mathcal{X} \leftarrow \mathrm{IFFT}_n(\mathcal{X}, D_n+1)$
  return $\mathcal{X}$

3.2. Efficient matrix-vector multiplication in the Chebyshev basis. An equally efficient matrix–vector multiplication algorithm with comparable asymptotic costs can be developed for Chebyshev systems. Unlike in the monomials, the codomain basis of the Macaulay operator (2.11) differs slightly from the domain: the domain uses the standard Chebyshev basis of the second kind, $\psi_k(x) = \cos(k \arccos x)$, whereas the codomain uses a modified basis $\phi_0(x) = 1/2$ and $\phi_k(x) = \psi_k(x)$ for $k \geq 1$. After homogenization, the bases transform to $\hat{\psi}_{d,k}(t,x) = t^d \cos\left(k \arccos(x/t)\right)$ and $\hat{\phi}_{d,k}(t,x) = t^d \cos\left(k \arccos(x/t)\right)$ for $k \geq 1$ and $\hat{\phi}_{d,0}(t,x) = \frac{1}{2}t^d$.

The scaling of the first Chebyshev term by one-half is intentional so that the Macaulay matrix attains a (multi-level) Toeplitz-plus-Hankel structure. When $N = 1$, the Macaulay matrix becomes a vertical concatenation of S Toeplitz-plus-Hankel matrices. For a single polynomial $g = \frac{c_0}{2}t^d + \sum_{k=1}^{d} c_k t^d \cos(d \arccos(x/t))$, the matrix

$\mathbf{M}(D;g) \in \mathbb{C}^{(D-d+1)\times(D+1)}$ takes on the form

$$\frac{1}{2}\left(\begin{bmatrix} c_0 & c_1 & \cdots & c_d & & & & \\ c_1 & c_0 & c_1 & \cdots & c_d & & & \\ \vdots & \ddots & \ddots & \ddots & & \ddots & & \\ c_d & \cdots & c_1 & c_0 & c_{1c} & \cdots & c_d & \\ & \ddots & & \ddots & \ddots & \ddots & & \ddots \\ & & c_d & \cdots & c_1 & c_0 & c_1 & \cdots & c_d \end{bmatrix} + \begin{bmatrix} c_0 & c_1 & \cdots & c_d & \\ c_1 & & \iddots & & \\ \vdots & \iddots & & & \\ c_d & & & & \\ & & & & \end{bmatrix}\right).$$

This structure is a direct consequence of Chebyshev identity

$$\hat{\psi}_{d,l}(x)\hat{\psi}_{d,k}(x) = \tfrac{1}{2}\left(\hat{\psi}_{d,l+k}(x) + \hat{\psi}_{d,|k-l|}(x)\right), \tag{3.3}$$

along with an analogous identity for $\hat{\phi}_{d,l}(x)$ that takes into account the one-half scaling. Similar to the monomial case, the matrix $\mathbf{M}(D;g)$ can be embedded into a larger $(D+1)\times(D+1)$ matrix that admits a decomposition involving trigonometric transforms. By setting $\boldsymbol{a} = (\boldsymbol{c}, \mathbf{0}_{D-d}) \in \mathbb{C}^{D+1}$, the top $D-d+1$ rows of a "Chebyshev"-circulant matrix $\mathrm{Chebcirc}(\boldsymbol{a}) \in \mathbb{C}^{D+1\times D+1}$ with entries

$$\frac{1}{2}\left(\begin{bmatrix} a_0 & a_1 & \cdots & a_{D-1} & a_D \\ a_1 & a_0 & a_1 & & a_{D-1} \\ \vdots & a_1 & a_0 & \ddots & \vdots \\ a_{D-1} & & \ddots & \ddots & a_1 \\ a_D & a_{D-1} & \cdots & a_1 & a_0 \end{bmatrix} + \begin{bmatrix} a_0 & a_1 & \cdots & a_{D-1} & a_D \\ a_1 & & a_{D-1} & a_D & 0 \\ \vdots & \iddots & a_D & 0 & \vdots \\ a_{D-1} & \iddots & & \iddots & -a_3 \\ a_D & 0 & \cdots & -a_3 & -a_2 \end{bmatrix}\right), \tag{3.4}$$

coincide with the matrix $\mathbf{M}(D;g)$. The following decomposition can be established:

$$\mathrm{Chebcirc}(\boldsymbol{a}) = \mathbf{S}_{D+1}\mathbf{C}_{D+1}\,\mathrm{diag}\left(\sqrt{\frac{D+1}{2}}\mathbf{C}_{D+1}^{\top}\mathbf{S}_{D+1}^{-1}\boldsymbol{a}\right)\mathbf{C}_{D+1}^{\top}\mathbf{S}_{D+1},$$

where $\mathbf{S}_{D+1} = \mathrm{diag}(\sqrt{2}, 1, \ldots, 1) \in \mathbb{R}^{D+1\times D+1}$, and

$$[\mathbf{C}_{D+1}]_{kl} = \sqrt{\frac{2-\delta_{k,1}}{D+1}}\cos\left(\tfrac{\pi}{D+1}\left(l-\tfrac{1}{2}\right)(k-1)\right), \quad \delta_{k,l} = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases},$$

is the DCT-II matrix [3]. As a result, left and right multiplication of $\mathbf{M}(D;g)$ with a vector can be efficiently carried out with the help of fast cosine transforms (FCTs).

For $N \geq 2$, the structure of the Macaulay matrix becomes increasingly intricate. Although the structure of the matrix still follows from the identity (3.3), the bookkeeping is tedious since the identity is distributed across each coordinate. For Chebyshev systems, the matrix $\mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}}) \in \mathbb{C}^{\sum_{s=1}^{S}\prod_{n=1}^{N}(D_n - d_n^{(s)}+1)\times\prod_{n=1}^{N}(D_n+1)}$ is a sum of 2^N terms:

$$\mathbf{M}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})) = \frac{1}{2^N}\sum_{w_1\in\{0,1\}}\cdots\sum_{w_N\in\{0,1\}}\mathbf{M}_{\boldsymbol{w}}(\boldsymbol{D};\Sigma_{\boldsymbol{t}}). \tag{3.5}$$

The entries of each term are specified by

$$[\mathbf{M}_{\boldsymbol{w}}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})]_{\imath(s,\boldsymbol{i}),\jmath(\boldsymbol{j})} = \begin{cases} c^{(s)}_{\boldsymbol{q}_{\boldsymbol{w}}(\boldsymbol{i},\boldsymbol{j})} & \text{if } \boldsymbol{q}_{\boldsymbol{w}}(\boldsymbol{i},\boldsymbol{j}) \in \mathscr{A}_N[\boldsymbol{d}^{(s)}], \\ 0 & \text{otherwise,} \end{cases} \tag{3.6}$$

where $\boldsymbol{q}_{\boldsymbol{w}}(\boldsymbol{i},\boldsymbol{j})$ is the element-wise application of the function

$$q_w(i,j) = \begin{cases} |j-i| & w=0, \\ i+j-2 & w=1. \end{cases}$$

The matrix $\mathbf{M}_{\boldsymbol{w}}(\boldsymbol{D};\Sigma_{\boldsymbol{t}})$ consists of a vertical concatenation of S multi-level Toeplitz-Hankel matrices $\mathbf{M}_{\boldsymbol{w}}(\boldsymbol{D};g^{(s)}) \in \mathbb{C}^{\prod_{n=1}^{N}(D_n-d_n^{(s)}+1)\times\prod_{n=1}^{N}(D_n+1)}$. The n-th entry of the subscript $\boldsymbol{w}$ determines the structure at the n-th level: if $w_n=0$, the matrix is Toeplitz at that level; if $w_n=1$, it is Hankel. For example, if $N=2$, then $\mathbf{M}(\boldsymbol{D};g^{(s)})$ is the average of four terms: a doubly Toeplitz $\mathbf{M}_{(0,0)}(\boldsymbol{D};g^{(s)})$, an inner-Toeplitz-outer-Hankel $\mathbf{M}_{(0,1)}(\boldsymbol{D};g^{(s)})$, inner-Hankel-outer-Toeplitz $\mathbf{M}_{(1,0)}(\boldsymbol{D};g^{(s)})$, and a doubly Hankel $\mathbf{M}_{(1,1)}(\boldsymbol{D};g^{(s)})$.

The embedding of $\mathbf{M}(\boldsymbol{D};\boldsymbol{g})$ into a higher-dimensional "Chebyshev"-circulant matrix proceeds as follows. By setting $\mathcal{A} \in \mathbb{C}^{(D+1)\times\ldots\times(D+1)}$ equal to $\mathcal{C}=\operatorname{coeff}(g)$ at $\mathcal{A}_{\mathbf{1}:(\boldsymbol{D}+\mathbf{1})}$ and zero elsewhere, $\mathbf{M}(\boldsymbol{D};\boldsymbol{g})$ can be embedded into the matrix

$$\operatorname{Chebcirc}(\mathcal{A}) = \frac{1}{2^N}\sum_{w_1\in\{0,1\}}\cdots\sum_{w_N\in\{0,1\}}\operatorname{Chebcirc}_{\boldsymbol{w}}(\mathcal{A}). \tag{3.7}$$

The entries of $\operatorname{Chebcirc}_{\boldsymbol{w}}(\mathcal{A}) \in \mathbb{C}^{\prod_{n=1}^{N}(D_n+1)\times\prod_{n=1}^{N}(D_n+1)}$ are specified by

$$[\operatorname{Chebcirc}_{\boldsymbol{w}}(\mathcal{A})]_{\jmath(\boldsymbol{i})\jmath(\boldsymbol{j})} = \chi(\mathcal{A};\boldsymbol{t}_{\boldsymbol{w}}(\boldsymbol{D},\boldsymbol{i},\boldsymbol{j}))\cos\left(\pi\left(\sum_{n=1}^{N} w_n H(i_n+j_n-2-D_n)\right)\right),$$

where $H(\cdot)$ is the Heavyside function, $\chi(\mathcal{A};\boldsymbol{i}) = a_{\boldsymbol{i}}$ if $\boldsymbol{i}\in\mathscr{A}_N[\boldsymbol{d}^{(s)}]$ and 0 otherwise, and $\boldsymbol{t}_{\boldsymbol{w}}(\boldsymbol{D},\boldsymbol{i},\boldsymbol{j})$ is the element-wise application of the function

$$t_w(D,i,j) = \begin{cases} |j-i| & w=0, \\ D+1-|i+j-D-3| & w=1. \end{cases}$$

Using the decomposition

$$\operatorname{Chebcirc}(\mathcal{A}) = \left(\bigotimes_{n=1}^{N}\hat{\mathbf{C}}_{D_n+1}\right)\operatorname{diag}\left(\left(\bigotimes_{N=1}^{N}\tilde{\mathbf{C}}_{D_n+1}\right)\operatorname{vec}(\mathcal{A})\right)\left(\bigotimes_{k=1}^{K}\hat{\mathbf{C}}_{D_n+1}^{\top}\right), \tag{3.8}$$

where $\hat{\mathbf{C}}_{D_n+1} = \mathbf{S}_{D_n+1}\mathbf{C}_{D_n+1}$ and $\tilde{\mathbf{C}}_{D_n+1} = \sqrt{\frac{D_n+1}{2}}\mathbf{C}_{D_n+1}^{\top}\mathbf{S}_{D_n+1}^{-1}$, efficient matrix-vector multiplication algorithms can be formulated utilizing FCTs. The procedure is detailed in Algorithm 3.2 and asymptotic flop and memory costs are shown in Table 1.

Algorithm 3.2 Right and left multiplication in the Chebyshev basis

function MV($\Sigma_{\boldsymbol{t}}$, $\boldsymbol{D}$, $\mathcal{X}$)
  for $n \in \{1, \ldots, N\}$ **do**
    $\mathcal{X} \leftarrow \mathcal{X} \cdot_n \mathbf{S}_{D_n+1}$
    $\mathcal{X} \leftarrow \mathrm{IFCT}_n(\mathcal{X}, D_n + 1)$
  for $s \in \{1, \ldots, S\}$ **do**
    for $n \in \{1, \ldots, N\}$ **do**
      $\mathcal{C}^{(s)} \leftarrow \mathcal{C}^{(s)} \cdot_n \mathbf{S}^{-1}_{d_n+1}$
      $\mathcal{C}^{(s)} \leftarrow (\frac{D_n+1}{2})^{\frac{1}{2}} \mathcal{C}^{(s)}$
      $\mathcal{C}^{(s)} \leftarrow \mathrm{IFCT}_n(\mathcal{C}^{(s)}, D_n+1)$
    $\mathcal{Y}^{(s)} \leftarrow \mathcal{X} * \mathcal{C}^{(s)}$
    for $n \in \{1, \ldots, N\}$ **do**
      $\mathcal{Y}^{(s)} \leftarrow \mathrm{FCT}_n(\mathcal{Y}^{(s)}, D_n+1)$
      $\mathcal{Y}^{(s)} \leftarrow \mathcal{Y}^{(s)} \cdot_n \mathbf{S}_{D_n+1}$
    $\mathcal{Y}^{(s)} \leftarrow \left[\mathcal{Y}^{(s)}\right]_{\mathbf{1}:(\boldsymbol{D}-\boldsymbol{d}^{(s)}+\mathbf{1})}$
  return $\left\{\mathcal{Y}^{(s)}\right\}_{s=1}^{S}$

function MVA($\Sigma_{\boldsymbol{t}}$, $\boldsymbol{D}$, $\{\mathcal{Y}^{(s)}\}_{s=1}^{S}$)
  Init zero $\mathcal{X} \in \mathbb{C}^{(D_1+1)\times\ldots\times(D_N+1)}$
  for $s \in \{1, \ldots, S\}$ **do**
    for $n \in \{1, \ldots, N\}$ **do**
      $\mathcal{C}^{(s)} \leftarrow \mathcal{C}^{(s)} \cdot_n \mathbf{S}^{-1}_{d_n+1}$
      $\mathcal{C}^{(s)} \leftarrow (\frac{D_n+1}{2})^{\frac{1}{2}} \mathcal{C}^{(s)}$
      $\mathcal{C}^{(s)} \leftarrow \mathrm{IFCT}_n(\mathcal{C}^{(s)}, D_n+1)$
      $\mathcal{Y}^{(s)} \leftarrow \mathcal{Y}^{(s)} \cdot_n \mathbf{S}_{D_n+1}$
      $\mathcal{Y}^{(s)} \leftarrow \mathrm{IFCT}_n(\mathcal{Y}^{(s)}, D_n+1)$
    $\mathcal{X} \leftarrow \mathcal{X} + \mathcal{Y}^{(s)} * \mathcal{C}^{(s)}$
  for $n \in \{1, \ldots, N\}$ **do**
    $\mathcal{X} \leftarrow \mathrm{FCT}_n(\mathcal{X}, D_n + 1)$
    $\mathcal{X} \leftarrow \mathcal{X} \cdot_n \mathbf{S}_{D_n+1}$
  return $\mathcal{X}$

Table 1: Asymptotic flop counts and memory requirements (in big-O notation) for right and left multiplication of the Macaulay matrix using various algorithms. For clarity, the estimates assume $d_n^{(s)} = d$, $D_n = D$, and $P^{(s)} = P$ for all $s \in 1, \ldots, S$ and $n \in 1, \ldots, N$. As shown in the table, the use of Algorithms 3.1 and 3.2 provides substantial computational savings in both flop and memory.

	Runtime (flop)	**Memory (no. of floats)**		
		$\mathbf{M}(\boldsymbol{D}; \Sigma_{\boldsymbol{t}})$	$\mathrm{vec}(\mathcal{X})$	$\{\mathrm{vec}(\mathcal{Y})\}_{s=1}^{S}$
Naive	SD^{2N}	$S(D-d)^N D^N$	D^N	$S(D-d)^N$
Alg. 3.1 & 3.2	$SND^N \log D$	Sd^N	D^N	$S(D-d)^N$

4. Application of Krylov techniques for null space computations. The Golub–Kahan bidiagonalization procedure is a two-sided Gram–Schmidt process that reduces a rectangular matrix to bidiagonal form using orthogonal transformations from both sides. The resulting orthogonal bases can be used to (i) solve least-squares problems or (ii) extract extremal singular triplets. The first applications leads to the LSQR method [39], while the second application leads to the implicitly restarted Lanczos bidiagonalization algorithm [6, 28]. We describe how these algorithms are used to compute a null space basis for a (numerically) rank-deficient matrix.

4.1. The LSQR approach. Given a rectangular matrix $\mathbf{M}$, a right-hand vector $\boldsymbol{b}$, and a starting vector $\boldsymbol{x}_0$, LSQR produces at the k-th iteration the solution

$$\boldsymbol{x}_k = \underset{\boldsymbol{\chi} \in \boldsymbol{x}_0 + \mathcal{K}_k(\mathbf{M}^\top \mathbf{M}, \boldsymbol{p}_1)}{\arg\min} \|\mathbf{M}\boldsymbol{\chi} - \boldsymbol{b}\|_2, \tag{4.1}$$

where $\boldsymbol{p}_1 = \mathbf{M}^\top \boldsymbol{q}_1$ and $\boldsymbol{q}_1 = (\boldsymbol{b} - \mathbf{M}\boldsymbol{x}_0)/\|\boldsymbol{b} - \mathbf{M}\boldsymbol{x}_0\|_2$. LSQR is mathematically equivalent to applying CG to the normal equations and thus $\boldsymbol{x}_k$ is the solution to minimization of the quadratic form $\phi(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^\top \mathbf{M}^\top \mathbf{M}\boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{b}$ in the affine space

$\boldsymbol{x}_0 + \mathcal{K}_k(\mathbf{M}^\top\mathbf{M}, \boldsymbol{b} - \mathbf{M}^\top\mathbf{M}\boldsymbol{x}_0)$. Despite this equivalence, LSQR has been shown to be numerically more robust than CG on the normal equations for ill-conditioned systems since it relies on the bidiagonalization process instead of tridiagonalization of the Gram matrix [39]. However, this robustness comes at the price of a slightly higher per-iteration cost [39].

For our problem, $\boldsymbol{b} = \mathbf{0}$ and $\mathbf{M} \in \mathbb{C}^{\sum_{s=1}^{S}\prod_{n=1}^{N}(D_n - d_n^{(s)}+1)\times\prod_{n=1}^{N}(D_n+1)}$ is the Macaulay matrix, which has an R-dimensional null space in the noiseless overdetermined case. Under these conditions, LSQR will find the orthogonal projection of $\boldsymbol{x}_0$ to $\operatorname{null}\mathbf{M}$, i.e., $\boldsymbol{x}_{0,n} := \operatorname{Proj}_{\operatorname{null}\mathbf{M}}\boldsymbol{x}_0$, in at most $\prod_{n=1}^{N}(D_n+1) - R$ steps. Since a randomly sampled unit vector typically has a nonzero component in the null space, LSQR can be used together with the fast matrix-vector products of section 3 to retrieve null vectors of the Macaulay matrix. While a single null space vector is sufficient to retrieve all roots of the system, Algorithm 4.1 describes a process to retrieve the entire null space. This is done by performing $\hat{R} \geq R$ parallel LSQR iterations with random starting vectors. By continuing the iterations until the normalized error

$$\boldsymbol{\epsilon}_k := \mathbf{M}\boldsymbol{x}_k / \|\boldsymbol{x}_k\| \tag{4.2}$$

falls below a threshold $\epsilon > 0$, converged vectors can be post-processed to compute an orthonormal basis. If at least R vectors converge within maxixum iteration bound $K_{\max}$, Algorithm 4.1 is expected to almost surely retrieve a null space basis with accuracy ϵ.

The complexity of the method depends on the number of iterations required to reach the error tolerance. This, in turn, depends on how quickly a vector in the Krylov subspace $\mathcal{K}_k(\mathbf{M}^\top\mathbf{M}, \boldsymbol{p}_1)$ can eliminate the component of $\boldsymbol{x}_0$ along the row space of $\mathbf{M}$. Using classical results [48, Theorem 38.5], the following bound can be produced:

$$\|\boldsymbol{x}_k - \boldsymbol{x}_{0,n}\|_2 \leq 2\kappa(\mathbf{M}) \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\boldsymbol{x}_0 - \boldsymbol{x}_{0,n}\|_2 ,$$

where $\kappa(\mathbf{M}) := \|\mathbf{M}\|_2 \|\mathbf{M}^\dagger\|_2$. However, faster convergence is possible if the singular values of $\mathbf{M}$ are clustered [48, Theorem 38.4]. In practice, the convergence behavior is far more difficult to analyze. Loss of conjugacy between directions and loss of orthogonality between residuals due to numerical rounding can significantly degrade the rate of descent, even to the point that stagnation occurs. In fact, the true solution may never be reached in a finite number of steps.

Nonetheless, for reasonably well-conditioned systems, convergence typically occurs within $C(\varepsilon)\prod_{n=1}^{N}(D_n+1)$ iterations for a small constant $C(\varepsilon) > 0$. In LSQR, the dominant cost stems from matrix–vector products, of which three are required per iteration[6]. Table 2 summarizes the complexity of Algorithm 4.1, assuming Algorithms 3.1 and 3.2 are used for the matrix–vector operations. Letting $\eta_{\text{LSQR}}(\varepsilon)$ denote the number of matrix–vector products needed to reach accuracy $\varepsilon > 0$, one obtains

$$\eta_{\text{LSQR}}(\varepsilon) \in \mathrm{O}(\hat{R}\, C(\varepsilon) D^N). \tag{4.3}$$

Thus, when $(D-d)^N/(\hat{R}C(\varepsilon)N\log D) \gg 1$, the flop saving over a full SVD will be substantial according to Table 2. The primary advantage, however, is memory efficiency: unlike the full SVD, algorithm 4.1 avoids storing full matrices. This is particularly beneficial when $\hat{R} = 1$, i.e., if a single null vector is used to retrieve the roots.

[6]Two are intrinsic to the algorithm, whereas one is required to evaluate (4.2).

Algorithm 4.1 Null space retrieval using the LSQR method

function NULL_LSQR($\mathbf{M}$, $\hat{R}$; ε, $K_{\max}$)
  for $r = 1, \dots, \hat{R}$ **do**
    Initialize random $\boldsymbol{x}_0 \in \mathbb{R}^{\prod_{n=1}^{N}(D_n+1)}$ with $\|\boldsymbol{x}_0\|_2 = 1$ and set $\boldsymbol{b} = \mathbf{0}$
    Run LSQR until $\|\boldsymbol{\epsilon}_k\| \leq \epsilon$ or if $K_{\max}$ iterations are reached
    if $\|\boldsymbol{\epsilon}_k\|_2 < \varepsilon$ **then**
      $\mathbf{X} \leftarrow \begin{bmatrix}\mathbf{X} & \boldsymbol{x}_k\end{bmatrix}$
  Compute orthonormal basis $\mathbf{K}$ of $\operatorname{col}\mathbf{X}$ ▷ Using SVD or QR.
  return K

A limitation of Algorithm 4.1 is its inability to handle noisy systems. Overdetermined systems with noise generally have only approximate roots, causing the Macaulay matrix to lose a strict null space. In such cases, LSQR eventually converges to the unique minimum of $\phi(\boldsymbol{x}) = \frac{1}{2}\|\mathbf{M}\boldsymbol{x}\|_2$ at $\boldsymbol{x} = \mathbf{0}$. One remedy is to minimize ϕ subject to the normalization constraint $\|\boldsymbol{x}\|_2 = 1$. More generally, the subspace spanned by the $\hat{R}$ smallest right-singular vectors is found by minimizing $\|\mathbf{M}\mathbf{X}\|_2^2$ for $\mathbf{X}$ on the $\hat{R}$-dimensional Stiefel manifold [1,11]. This approach, however, departs from a purely Krylov-based method.

4.2. Implicitly restarted Lanczos bidiagonalization. An alternative, which stays within the Krylov-subspace framework, is to approximate the singular triplets of $\mathbf{M}$ directly using the Lanczos bidiagonalization method. At the k-th step, the right-sided Golub–Kahan process produces the matrices $\mathbf{P}_k = \begin{bmatrix}\boldsymbol{p}_1 & \cdots & \boldsymbol{p}_k\end{bmatrix}$ and $\mathbf{Q}_k = \begin{bmatrix}\boldsymbol{q}_1 & \cdots & \boldsymbol{q}_k\end{bmatrix}$, whose columns span the Krylov subspaces $\mathscr{K}_k(\mathbf{M}^\top\mathbf{M}, \boldsymbol{p}_1)$ and $\mathscr{K}_k(\mathbf{M}\mathbf{M}^\top, \mathbf{M}\boldsymbol{p}_1)$, respectively. Upon termination at this step, the SVD variant of the Lanczos method computes the singular triplets $(\hat{\sigma}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}})$ of the bidiagonal matrix $\mathbf{B}_k = \mathbf{Q}_k^\top\mathbf{M}\mathbf{P}_k$. These, in turn, yield the Ritz triples $(\hat{\sigma}, \mathbf{Q}_k\hat{\boldsymbol{u}}, \mathbf{P}_k\hat{\boldsymbol{v}})$, which provide in a certain sense the best approximations of true singular triplets within the computed Krylov subspaces. In the case of right-singular vectors, the accuracy of the approximations is fully determined by the subspace captured by $\mathscr{K}_k(\mathbf{M}^\top\mathbf{M}, \boldsymbol{p}_1)$. In particular, when the matrix $\mathbf{M}$ has an exact null space, the computation of a null vector depends entirely on $\boldsymbol{p}_1$ having a nonzero component in the null space and how the subsequent vectors in $\mathbf{P}_k$ are able to cancel the component of $\boldsymbol{p}_1$ along the row space of $\mathbf{M}$. From this perspective, the retrieval of a null space vector follows the same mechanism as LSQR, and a comparable convergence rate can be expected.

Unfortunately, the Lanczos bidiagonalization method, in its standard form, is not a practical algorithm. Unlike LSQR, the Lanczos method requires explicit storage of $\mathbf{P}_k$ and $\mathbf{Q}_k$. Thus, unless the singular values of $\mathbf{M}$ are highly clustered, the number of Golub–Kahan steps required to obtain accurate approximations will be prohibitively large. Moreover, maintaining orthogonality of $\mathbf{P}_k$ and $\mathbf{Q}_k$ is far more critical because loss of orthogonality can lead to instability and spurious singular values. The additional cost of re-orthogonalization further undermines the practicality of the method. The remedy is to stop the Golub–Kahan procedure prematurely and use the available spectral information to restart the process. A powerful restarting technique is *implicit restarting*, which replaces the initial vector $\boldsymbol{p}_1$ with $\tilde{\boldsymbol{p}}_1 = \psi(\mathbf{M}^\top\mathbf{M})\boldsymbol{p}_1$, where $\psi(x) = (x - \mu_1^2)\cdots(x - \mu_l^2)$ is a polynomial filter that removes unwanted spectral components. Since the right-singular vectors associated with the smallest singular values are of interest, an effective choice is to set $\mu_1, \dots, \mu_l$ equal to the l largest

computed Ritz values. Repeated restarts then progressively steer the starting vector toward the subspace spanned by the smallest singular values, enabling recovery of a single null vector. For computing the smallest singular triplets, implicit restarting was first introduced in [28], where harmonic Ritz values [42] were employed as shifts to accelerate convergence, particularly when the smallest singular values are tightly clustered. Subsequently, [6] proposed a numerically more stable variant that performs restarting by augmenting the Krylov subspace with previously computed Ritz vectors. The authors also suggested using harmonic Ritz vectors for augmentation, though this requires $\mathbf{B}_k^{-1}$ and may become unstable if $\mathbf{B}_k$ is ill-conditioned.

To compute an R-dimensional null space, a block Lanczos variant with block size $\hat{R} \geq R$ is required. Algorithm 4.2 outlines the augmented implicitly restarted block Lanczos algorithm (IRLBA); see [6, 7] for details. Let η_{IRLBA} denote the number of matrix–vector products required in Algorithm 4.2 to achieve accuracy $\varepsilon > 0$. Table 2 summarizes the asymptotic flop and memory costs of IRLBA, including the reorthogonalization overhead. The total number of matrix–vector products satisfies

$$\eta_{\text{IRLBA}}(\varepsilon) \in \mathrm{O}(k\hat{R}T(\varepsilon)),$$

where $T(\varepsilon)$ is the number of restarts. For IRLBA to perform well, careful tuning of its hyperparameters is required. Increasing the block size $\hat{R}$ and the restarting window k generally improves robustness and convergence, but at the cost of increased memory usage and more expensive re-orthogonalization operations. Furthermore, restarting typically slows down the convergence [34]. In contrast, LSQR does not employ restarting; however, the absence of re-orthogonalization can likewise have a detrimental effect on convergence.

Algorithm 4.2 Null space retrieval using the IRLBA method

function NULL_IRLBA($\mathbf{M}$, $\hat{R}$; ε, k, $T_{\max}$)
  Initialize random orthonormal matrix $\mathbf{P}_{\hat{R}} \in \mathbb{R}^{\prod_{n=1}^{N}(D_n+1)\times\hat{R}}$.
  Construct $\mathbf{P}_{k\hat{R}}$, $\mathbf{Q}_{k\hat{R}}$, $\mathbf{Q}_{k\hat{R}}$, and $\mathbf{B}_{k\hat{R}}$ using the Golub–Kahan process.
  for $t = 1, \ldots, T_{\max}$ **do**
    $\hat{\mathbf{U}}, \hat{\mathbf{\Sigma}}, \hat{\mathbf{V}} \leftarrow \mathrm{SVD}(\mathbf{B}_{k\hat{R}})$
    if $\|\mathbf{M}(\mathbf{P}_{\hat{R}k}\hat{\boldsymbol{u}}_i) - \tilde{\sigma}_i(\mathbf{Q}_{\hat{R}k}\hat{\boldsymbol{u}}_i)\| < \varepsilon$ for $i \in \{\hat{R}(k-1)+1, \ldots, \hat{R}k\}$ **then**
      break
    Restart Lanczos by augmenting with (harmonic) Ritz vectors
    Produce $\mathbf{P}_{k\hat{R}}$, $\mathbf{Q}_{k\hat{R}}$, $\mathbf{Q}_{k\hat{R}}$, and $\mathbf{B}_{k\hat{R}}$ after k steps Golub–Kahan process
  $\mathbf{K} := \mathbf{P}_{\hat{R}k}\begin{bmatrix}\tilde{\boldsymbol{v}}_\tau & \cdots & \tilde{\boldsymbol{v}}_{\hat{R}k}\end{bmatrix}$
  return $\mathbf{K}$

5. Numerical experiments. In this section, we evaluate the practical performance of utilizing the fast matrix-vector products from section 3 in Algorithms 4.1 and 4.2. The performances are compared to the baseline SVD-based method. We first consider overdetermined systems in the noise-free setting and subsequently study performance under noisy conditions. Before comparing the null space algorithms, however, we establish key insights into how root estimation and null space computation depend on the number of available null space vectors and the number of equations.

5.1. Experimental setup. We outline how test problems are generated, define error metrics for evaluating performance, and specify the computational environment.

Table 2: Asymptotic complexities (in big-O notation) of LSQR (Algorithm 4.1) and IRLBA (Algorithm 4.2) in terms of flop count and memory. We assume $d_n^{(s)} = d$ and $D_n = D$. The SVD algorithm refers to either Golub–Reinsch or R-SVD [24], which have similar asymptotic complexity. The estimates show that, when the number of matrix–vector products remains small, LSQR and IRLBA outperform the SVD in terms of flop count while simultaneously requiring significantly less memory.

	Runtime (flop)	**Memory (no. of floats)**
SVD	$\left((D-d)^N S\right) D^{2N}$	$S(D-d)^N D^N + D^{2N}$
LSQR	$\left(\hat{R}\, C(\varepsilon) N S \log D\right) D^{2N}$	$Sd^N + \hat{R}(D^N + S(D-d)^N)$
IRLBA	$\left((NS \log D + k\hat{R}) k\hat{R} T(\epsilon)\right) D^{2N}$	$Sd^N + k\hat{R}(D^N + S(D-d)^N)$

Generation of test problems. The algorithms are tested on randomly generated overdetermined polynomial systems with R roots at prescribed locations. We limit our focus on systems with real coefficients and real roots, although the theory in section 2 does not impose any such restriction. Random overdetermined systems are constructed as follows. The null space of a multivariate Vandermonde matrix evaluated at prescribed points defines a subspace of polynomials that vanish at these points. By selecting $S > N$ random elements from this subspace, one obtains, with probability one, an overdetermined system with exactly R common roots. Gaussian noise can be added a posteriori to the polynomial coefficients to produce systems with only approximate solutions. To quantify this, we define the signal-to-noise ratio (SNR) as the ratio of the norm of the polynomial coefficients to the norm of the added noise.

Error metrics. The SVD-based method and Algorithms 4.1 and 4.2 compute an orthonormal basis $\mathbf{K}$ for the (approximate) null space. Let $\mathbf{K}_{\text{ref}}$ denote the matrix whose columns are the right-singular vectors corresponding to the R smallest singular values of the Macaulay matrix. The following error metrics

$$E_1 = \frac{\|\mathbf{M}(\Sigma_{\boldsymbol{t}}, \boldsymbol{D})\mathbf{K}\|_2}{\|\mathbf{M}(\Sigma_{\boldsymbol{t}}, \boldsymbol{D})\|_2} \quad \text{and} \quad E_2 = \|(\mathbf{I} - \mathbf{K}_{\text{ref}}\mathbf{K}_{\text{ref}}^\top)\mathbf{K}\|_2 \tag{5.1}$$

are used to assess the accuracy of the computed null spaces. When the Macaulay matrix admits an exact null space, E_1 is used. For noisy systems, where the Macaulay matrix only possesses an approximate null space, the second metric is more appropriate. Specifically, E_2 measures the sine of the largest principal angle between the spaces defined by $\mathbf{K}$ and $\mathbf{K}_{\text{ref}}$. The accuracy of the estimated roots $\{\hat{\boldsymbol{x}}^{(r)}\}_{r=1}^R$ relative to the exact roots of unperturbed noiseless system $\{\boldsymbol{x}^{(r)}\}_{r=1}^R$ is estimated with the root-mean-squared (RMS) error

$$E_3 = \min_{\sigma \in \mathscr{P}_R} \sqrt{\frac{1}{R} \sum_{r=1}^{R} \|\boldsymbol{x}^{(r)} - \hat{\boldsymbol{x}}^{(\sigma(r))}\|_2^2}, \tag{5.2}$$

where the optimal assignment σ is computed using the Munkres algorithm over the set of all permutations $\mathscr{P}_R$.

Hardware & software environment. All experiments were conducted in MATLAB 2023a on a MacBook with an Apple M1 CPU (8 cores) and 16 GB of RAM. To

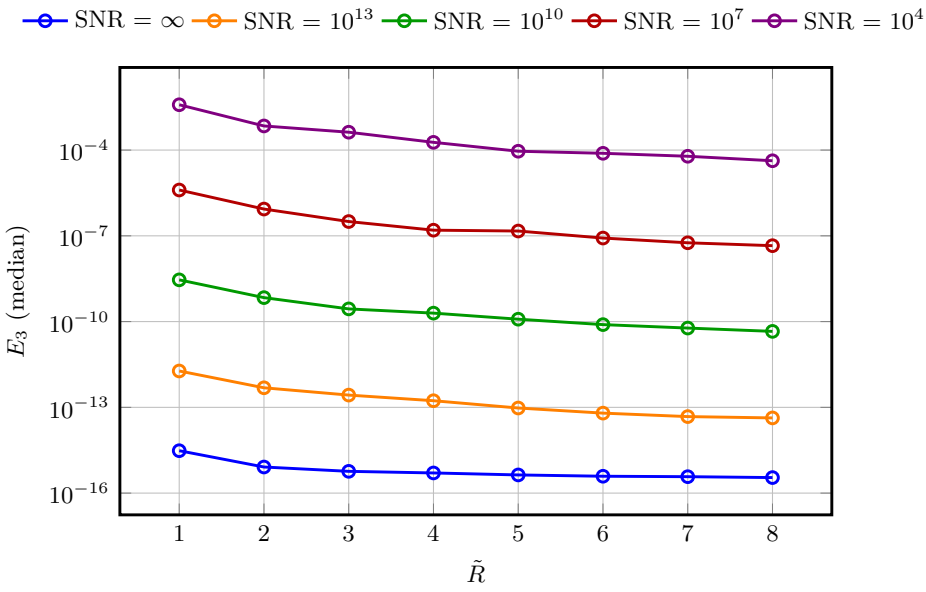


Fig. 1: Root estimation error (5.2) versus the number of null space vectors $\tilde{R}$ used in the CPD of (2.16) for various signal-to-noise ratios. Medians over 40 trials are shown for overdetermined systems ($N = 3$, $\boldsymbol{d} = (3, 3, 3)$) in the Chebyshev basis, with $R = 7$ roots uniformly sampled from $[-1, 1]^N$. The `cpd` function from Tensorlab [52] was used with settings `MaxIter = 400`, `TolFun = 1e-30`, and `TolX = 5e-16`. The approximate null space for constructing $\mathcal{T}$ is extracted from a multi-degree $\boldsymbol{D} = (8, 8, 8)$ Macaulay matrix using the $\tilde{R}$ smallest right singular vectors from the SVD.

compute the CPD in (2.16), we utilized the built-in `cpd` function of Tensorlab [52]. For algorithm 4.2, we employed the `irlbablk` implementation available on James Baglama's website[7].

5.2. Root estimation and available null space vectors. As discussed in subsection 2.7, the CPD in (2.16) can already be unique for a single null space vector $\tilde{R} = 1$. In practice, however, Figure 1 reveals that using more Macaulay null space vectors is numerically advantageous. The figure reports the median root estimation error E_3 for overdetermined systems with $R = 7$ roots as a function of the number of null space vectors. For all noise levels, the same trend is observed: increasing the number of null space vectors consistently reduces the error. However, for high SNR systems, most of the improvement is already achieved in the inclusion of one or two additional null space vectors. This indicates that a full null space basis is not always necessary for accurate root estimation.

5.3. Effect of number of equations on null space estimation. Experiments on random overdetermined systems show that the number of equations S affects null space estimation in two ways. First, increasing S lowers the multi-degree at which the Macaulay matrix's nullity stabilizes (see Table 3), allowing system roots to be estimated from smaller matrices. Second, larger S values yield a more favorable sin-

[7] https://www.math.uri.edu/~jbaglama/

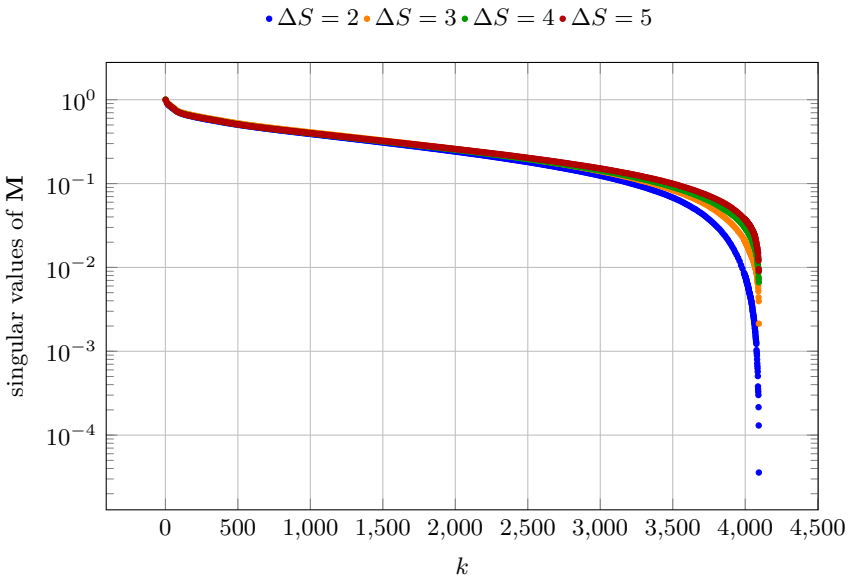
Table 3: Nullity of the Macaulay matrix as a function of the multi-degree $\boldsymbol{D} = D(1,1,1)$ and the number of extra equations $\Delta S = S - N$. We consider overdetermined systems with multi-degree $\boldsymbol{d} = (3,3,3)$ in the Chebyshev basis, with $N = 3$ variables and $R = 7$ roots. Adding consistent equations reduces the multi-degree at which the stabilization degree is reached. The case $\Delta S = 0$ is excluded, since for square systems Bernstein–Kushnirenko's theorem [23, 40] predicts 162 roots rather than 7.

D	$\Delta S = 1$	$\Delta S = 2$	$\Delta S = 3$	$\Delta S = 4$	$\Delta S = 5$	$\Delta S = 6$
3	60	59	58	57	56	55
4	93	85	77	69	61	53
5	108	81	54	27	7	7
6	93	33	7	7	7	7
7	60	7	7	7	7	7
8	27	7	7	7	7	7
9	8	7	7	7	7	7

gular value distribution for Krylov methods, with a sharper tail and smaller condition numbers. Figure 2 illustrates this observation for Macaulay matrices of a system with $N = 3$ variables. The figure also shows the null space error E_1 at different stages of algorithm 4.1, demonstrating that increasing S reduces the number of LSQR iterations needed to reach machine-precision accuracy.

5.4. Null space estimation in noise-free conditions. In the noise-free setting, the Macaulay matrix has a proper null space whenever the overdetermined system admits a positive number of isolated roots. Table 4 provides an indicative assessment of the performance of Algorithms 4.1 and 4.2 in regimes where the number of roots R is far smaller than the theoretical limits for square systems. In Table 4, we fix $R = 2$ and consider overdetermined systems with an increasing number of variables N, while keeping the coordinate degree equal to one. The table reports the average computation time and accuracy obtained with the proposed Krylov methods and with the SVD for computing the two-dimensional null space of the Macaulay matrix at the stabilization degree. Initially, the SVD method outperforms the Krylov approaches due to its highly optimized implementation; however, as N increases, Algorithms 4.1 and 4.2 eventually compute the null space more efficiently while maintaining satisfactory accuracy. Beyond a certain problem size, the SVD approach becomes infeasible due to insufficient memory to form the large dense matrices. In Table 4, the Krylov methods are driven to near machine precision to allow a fair comparison. In practice, this level of accuracy is often unnecessary, as relatively inexpensive Newton iterations can refine the solution obtained from a less accurate null space computation.

5.5. Null space estimation in noisy conditions. For noisy systems, the Macaulay matrix has an approximate null space. In this setting, the goal is to recover the subspace spanned by the right-singular vectors corresponding to the smallest singular values. Tables 5 and 6 provide an indicative assessment of IRLBA's ability to compute this subspace relative to the SVD method. Specifically, Table 5 evaluates IRLBA's performance for systems of increasing degree, while Table 6 examines how noise impacts the computation of the approximate null space. Similar to the noiseless case, IRLBA eventually outperforms the SVD in computation time while the errors



ΔS	$\kappa(\mathbf{M})$	E_1 after k iterations (normalized with number of columns)				
		0	0.125	0.25	0.5	1
2	27948	0.16484	5.8185e-05	1.8646e-05	9.1522e-06	2.1172e-06
3	468	0.16868	1.538e-05	7.9785e-07	1.2705e-11	1.0577e-15
4	150	0.17006	7.9003e-07	1.8766e-10	7.6271e-16	7.6087e-16
5	110	0.1629	5.6179e-08	3.8833e-14	6.5022e-16	1.5487e-15

Fig. 2: Singular value distribution of Macaulay matrices as a function of the number of additional equations, $\Delta S = S - N$. The overdetermined polynomial system is in the monomial basis with $N = 6$ variables, multi-degree $\boldsymbol{d} = (1, 1, 1, 1, 1, 1)$, and $R = 2$ roots. The Macaulay matrix is constructed at multi-degree $\boldsymbol{D} = (3, 3, 3, 3, 3, 3)$. Adding more equations sharpens the tail of the distribution and reduces the condition number $\kappa(\mathbf{M}) := \|\mathbf{M}\|_2 \, \|\mathbf{M}^\dagger\|_2$, resulting in faster convergence of algorithm 4.1 to machine-precision accuracy.

remain within an acceptable range.

6. Conclusions. We introduced two Krylov-based methods, derived from the Golub–Kahan bidiagonalization process, for computing an approximate null space basis of the Macaulay matrix. Together with the fast matrix-vector products derived for Macaulay matrices in the monomial and Chebyshev basis, these methods show promise of being significantly more efficient in both memory and computational time than the baseline SVD approach for overdetermined systems with a small number of roots. This potential is further supported by numerical experiments showing that a complete null space basis is not required to obtain reasonably accurate root estimates. Nevertheless, the potential efficiency gains are contingent on the Macaulay matrix exhibiting a favorable singular value distribution that accelerates the convergence of the Krylov methods. Our experiments indicate that, at least for the random systems considered, the singular value spectrum improves as the number of additional equations

Table 4: Median computation time and accuracy over 30 trials for computing the null space of the Macaulay matrix in a noise-free setting using LSQR, IRLBA, and the SVD. We consider overdetermined systems with $S = 2N$ equations of coordinate degree $d_i = 1$ in the monomial basis having $R = 2$ isolated roots at $(1, \ldots, 1)$ and $(-1, \ldots, -1)$. The Macaulay matrix is constructed at coordinate degree $D_i = 3$. For the Krylov methods, we set $\hat{R} = R = 2$ and $\epsilon = \|\mathbf{M}\|_2 \cdot 10^{-15}$. For IRLBA, we terminate the restarts explicitly when the second-to-last singular value reaches ϵ. Furthermore, we have optimized the window size k through manual experimentation. As the number of variables N increases, the Krylov methods eventually outperform the SVD in efficiency.

N	size $\mathbf{M}$		computation time (s)			E_1		
	rows	cols	SVD	LSQR	IRLBA	SVD	LSQR	IRLBA
2	36	16	1.3e-04	5.4e-03	1.7e-02	3.2e-16	1.3e-15	1.0e-13
3	162	64	2.7e-03	2.2e-02	3.4e-02	5.1e-16	4.2e-15	2.7e-15
4	648	256	3.0e-02	1.0e-01	1.8e-01	7.5e-16	9.4e-15	1.9e-15
5	2430	1024	7.3e-01	5.9e-01	9.9e-01	9.6e-16	3.6e-14	1.1e-14
6	8748	4096	3.4e+01	7.8e+00	9.1e+00	1.7e-15	4.0e-14	1.7e-14
7	30618	16384	–	5.8e+01	8.0e+01	–	9.9e-14	6.9e-13

Table 5: Median computation time and accuracy over 30 trials for computing an approximate null space of the Macaulay matrix in noisy conditions (SNR $= 10^8$) using IRLBA and SVD. In contrast to the experiment in Table 4, the multi-degree $\boldsymbol{d} = (d, d, d)$ is varied while $N = 3$, $S = 13$, $R = 3$, and $\boldsymbol{D} = (2d, 2d, 2d)$. The error on the IRLBA-computed null space E_2 is evaluated by using the null space from the SVD as the reference. As the problem size increases, IRLBA eventually outperforms the SVD in terms of computation time.

d	size $\mathbf{M}$		computation time (s)		E_2
	rows	cols	SVD	IRLBA	
2	351	125	1.6e-02	6.3e-02	5.0e-15
3	832	343	8.4e-02	2.6e-01	7.8e-14
4	1625	729	3.5e-01	9.6e-01	6.7e-13
5	2808	1331	1.4e+00	1.6e+00	7.8e-13
6	4459	2197	4.9e+00	3.4e+00	1.8e-12
7	6656	3375	1.6e+01	7.0e+00	8.5e-12
8	9477	4913	7.0e+01	1.6e+01	1.5e-11
9	13000	6859	1.7e+02	2.7e+01	4.7e-11

increases, leading to fast convergence even without preconditioners.

7. Future work. Our work does not explore the full range of methods available in the extensive Krylov and iterative methods literature, as this lies beyond the scope of the present study. While broader exploration may improve performance, the matrix–vector product scales exponentially with the number of variables, limiting the solvability of larger systems. We therefore argue that future efforts are better focused on overdetermined systems with additional structure, such as low-rank properties,

Table 6: Median computation time and accuracy over 30 trials for computing an approximate null space of the Macaulay matrix in noisy conditions using IRLBA and SVD. We vary the SNR for an overdetermined system with $N = 3$, $S = 9$, $R = 7$, $\boldsymbol{d} = (3, 3, 3)$, and $\boldsymbol{D} = (8, 8, 8)$.

SNR	computation time (s)		E_2
	SVD	IRLBA	
∞	1.6e+00	3.1e+00	7.6e-13
1.0e+13	1.9e+00	2.9e+00	4.9e-13
1.0e+10	1.7e+00	2.4e+00	5.3e-13
1.0e+7	1.7e+00	1.8e+00	6.4e-13
1.0e+4	1.7e+00	1.4e+00	9.5e-12

where this curse of dimensionality is alleviated. Another direction is the integration of Krylov techniques within the recursive frameworks [8,51]. Finally, investigating effective preconditioners for the Macaulay matrix to address more challenging problems is a worthwhile direction for future research.

Acknowledgments. We would like to thank Lothar Reichel, James Baglama, and Nicola Mastronardi for their feedback and useful suggestions. The authors also acknowledge the use of generative AI tools for language refinement in this work.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2008.
[2] W. W. Adams and P. Loustaunau, *An introduction to Grobner bases*, vol. 3, American Mathematical Soc., 1994.
[3] N. Ahmed, T. Natarajan, and K. R. Rao, *Discrete cosine transform*, IEEE Trans. Comput., 100 (2006), pp. 90–93.
[4] M. F. Atiyah and I. G. Macdonald, *Introduction to commutative algebra*, CRC Press, 2018.
[5] W. Auzinger and H. J. Stetter, *An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations*, in Numerical Mathematics, R. Agarwal, Y. Chow, and S. Wilson, eds., vol. 86 of International Series of Numerical Mathematics, Birkhäuser, Basel, Switzerland, 1988.
[6] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.
[7] J. Baglama and L. Reichel, *Restarted block lanczos bidiagonalization methods*, Numer. Algorithms, 43 (2006), pp. 251–272.
[8] K. Batselier, P. Dreesen, and B. De Moor, *A fast recursive orthogonalization scheme for the Macaulay matrix*, J. Comput. Appl. Math., 267 (2014), pp. 20–32.
[9] M. R. Bender and S. Telen, *Toric eigenvalue methods for solving sparse polynomial systems*, Math. Comput., 91 (2022), pp. 2397–2429.
[10] W. Boege, R. Gebauer, and H. Kredel, *Some examples for solving systems of algebraic equations by calculating Groebner bases*, J. Symb. Comput., 2 (1986), pp. 83–98.
[11] N. Boumal, *An introduction to optimization on smooth manifolds*, Cambridge University Press, 2023.
[12] J.-P. Cardinal, *Solving square polynomial systems: a practical method using Bezout matrices*, July 2018. arXiv:1807.11088 [math].
[13] R. M. Corless, P. M. Gianni, and B. M. Trager, *A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots*, in Proc. of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC '97, New York, NY, USA, 1997, Association for Computing Machinery, p. 133–140.
[14] D. A. Cox, *Applications of polynomial systems*, vol. 134, American Mathematical Soc., 2020.

[15] D. A. Cox, *Stickelberger and the eigenvalue theorem*, in Commutative Algebra, Springer, 2021, pp. 283–298.
[16] D. A. Cox, J. B. Little, and D. O'Shea, *Using algebraic geometry*, vol. 185 of Graduate Texts in Mathematics, Springer, New York, NY, USA, 1998.
[17] D. A. Cox, J. B. Little, and D. O'Shea, *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*, Springer, New York, NY, USA, 2013.
[18] A. Dickenstein and I. Z. Emiris, *Solving polynomial equations: Foundations, algorithms, and applications*, Springer, 2005.
[19] I. Domanov and L. De Lathauwer, *Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm*, Linear Algebra Appl., 513 (2017), pp. 342–375.
[20] P. Dreesen, *Back to the roots: Polynomial system solving using linear algebra*, PhD thesis, KU Leuven, Leuven, Belgium, 2013.
[21] D. Eisenbud, *Commutative algebra: With a view toward algebraic geometry*, vol. 150, Springer Science & Business Media, 2013.
[22] I. Z. Emiris and B. Mourrain, *Matrices in elimination theory*, J. Symb. Comput., 28 (1999), pp. 3–44.
[23] W. Fulton, *Introduction to toric varieties*, no. 131 in Ann. of Math. Stud., Princeton University Press, 1993.
[24] G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, MD, USA, 4 ed., 2012.
[25] N. Govindarajan, R. Widdershoven, S. Chandrasekaran, and L. De Lathauwer, *A fast algorithm for computing macaulay null spaces of bivariate polynomial systems*, SIAM J. Matrix Anal. Appl., 45 (2024), pp. 368–396.
[26] J. Harris, *Algebraic geometry: A first course*, vol. 133, Springer Science & Business Media, 2013.
[27] G. Jónsson and S. Vavasis, *Accurate solution of polynomial equations using Macaulay resultant matrices*, Math. Comput., 74 (2005), pp. 221–262.
[28] E. Kokiopoulou, C. Bekas, and E. Gallopoulos, *Computing smallest singular triplets with implicitly restarted lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39–61.
[29] D. Lazard, *Résolution des systemes d'équations algébriques*, Theor. Comput. Sci., 15 (1981), pp. 77–110.
[30] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, SIAM, 1998.
[31] T. Li, *Numerical solution of polynomial systems by homotopy continuation methods*, Handb. Numer. Anal., 11 (2003), pp. 209–304.
[32] D. Maclagan and G. G. Smith, *Uniform bounds on multigraded regularity*, arXiv preprint math/0305215, (2003).
[33] A. Morgan, *Solving polynomial systems using continuation for engineering and scientific problems*, SIAM, 2009.
[34] R. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comput., 65 (1996), pp. 1213–1230.
[35] B. Mourrain, *A new criterion for normal form algorithms*, in Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, M. Fossorier, H. Imai, S. Lin, and A. Poli, eds., vol. 1719 of Lecture Notes in Computer Science, Springer Verlag, 1999.
[36] B. Mourrain, *Bezoutian and quotient ring structure*, J. Symb. Comput., 39 (2005), pp. 397–415.
[37] B. Mourrain, S. Telen, and M. Van Barel, *Truncated normal forms for solving polynomial systems: Generalized and efficient algorithms*, J. Symb. Comput., 102 (2021), pp. 63–85.
[38] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Computing the common zeros of two bivariate functions via Bézout resultants*, Numer. Math., 129 (2015), pp. 181–209.
[39] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.
[40] I. R. Shafarevich and M. Reid, *Basic algebraic geometry*, vol. 2, Springer, 1994.
[41] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, *Tensor decomposition for signal processing and machine learning*, IEEE Transactions on Signal Processing, 65 (2017), pp. 3551–3582.
[42] G. L. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Rev., 42 (2000), pp. 267–293.
[43] A. J. Sommese and C. W. Wampler, *The numerical solution of systems of polynomials: Arising in engineering and science.*, World Scientific, 2005.

[44] H. J. Stetter, *Matrix eigenproblems are at the heart of polynomial system solving*, SIGSAM Bull., 30 (1996), p. 22–25.
[45] H. J. Stetter, *Numerical polynomial algebra*, SIAM, Philadelphia, PA, USA, 2004.
[46] B. Sturmfels, *Solving systems of polynomial equations*, no. 97 in CBMS Regional Conference Series in Mathematics, American Mathematical Soc., 2002.
[47] S. Telen, B. Mourrain, and M. Van Barel, *Solving polynomial systems via truncated normal forms*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1421–1447.
[48] L. N. Trefethen and D. Bau, *Numerical linear algebra*, vol. 50, SIAM, 1997.
[49] J. Vanderstukken and L. De Lathauwer, *Systems of polynomial equations, higher-order tensor decompositions and multidimensional harmonic retrieval: A unifying framework. Part I: The canonical polyadic decomposition*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 883–912.
[50] J. Vanderstukken, P. Kürschner, I. Domanov, and L. De Lathauwer, *Systems of polynomial equations, higher-order tensor decompositions, and multidimensional harmonic retrieval: A unifying framework. Part II: The block term decomposition*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 913–953.
[51] C. Vermeersch and B. De Moor, *Recursive algorithms to update a numerical basis matrix of the null space of the block row, (banded) block Toeplitz, and block Macaulay matrix*, SIAM J. Sci. Comput., 45 (2023), pp. A596–A620.
[52] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, *Tensorlab 3.0*, Mar. 2016, https://www.tensorlab.net. Available online.
[53] R. Widdershoven, N. Govindarajan, and L. De Lathauwer, *Overdetermined systems of polynomial equations: Tensor-based solution and application*, in Proc. of the 31st European Signal Processing Conference (EUSIPCO), Helsinki, Finland, September 2023, pp. 650–654.
[54] R. Widdershoven, N. Govindarajan, and L. De Lathauwer, *Fast Macaulay null space through the intersection of shifted null spaces*, Tech. Report 25-56, ESAT-STADIUS, KU Leuven, Leuven, Belgium, 2025.