A FAST ALGORITHM FOR COMPUTING MACAULAY NULL SPACES OF BIVARIATE POLYNOMIAL SYSTEMS*

NITHIN GOVINDARAJAN†, RAPHAËL WIDDERSHOVEN†, SHIVKUMAR CHANDRASEKARAN‡, AND LIEVEN DE LATHAUWER†

Abstract. As a crucial first step towards finding the (approximate) common roots of a (possibly overdetermined) bivariate polynomial system of equations, the problem of determining an explicit numerical basis for the right null space of the system's Macaulay matrix is considered. If $d_\Sigma \in \mathbb{N}$ denotes the total degree of the bivariate polynomials of the system, the cost of computing a null space basis containing all system roots is $\mathcal{O}(d_\Sigma^6)$ floating point operations through standard numerical algebra techniques (e.g., a singular value decomposition, rank-revealing QR-decomposition). We show that it is actually possible to design an algorithm that reduces the complexity to $\mathcal{O}(d_\Sigma^5)$. The proposed algorithm exploits the Toeplitz structures of the Macaulay matrix under a non-graded lexicographic ordering of its entries and uses the low displacement rank properties to efficiently convert it into a Cauchy-like matrix with the help of fast Fourier transforms. By modifying the classical Schur algorithm with total pivoting for Cauchy-like matrices, a compact representation of the right null space is eventually obtained from a rank-revealing LU-factorization. Details of the proposed method, including numerical experiments, are fully provided for the case wherein the polynomials are expressed in the monomial basis. Furthermore, it is shown that an analogous fast algorithm can also be formulated for polynomial systems expressed in the Chebyshev basis.

Key words. Macaulay matrices, polynomials systems, rank-revealing LU-factorizations, low displacement rank matrices, Schur algorithm.

AMS subject classifications. 15A69, 15A23

1. Introduction.

Solving systems of multivariate polynomial equations is a classical problem in mathematics. While degenerate cases of this problem, such as linear systems and univariate polynomial root-solving, have evolved into separate disciplines of their own, the more general case has been thoroughly studied in the field of (computational) algebraic geometry [13,14]. In circumstances where the system of polynomial equations only admits a finite number of solutions, i.e., so-called zero-dimensional systems, the literature has advocated two major approaches to find all common roots. The *first approach*, which effectively only applies to square systems, employs homotopy continuation to retrieve the roots of the desired system by continuous deformation of a "starting system" for which the roots are already known [2, 33, 46, 54, 55]. The *second approach*, which is more in line with the focus of this paper, are algebraic methods [1, 19, 32, 48, 49].

The goal in algebraic methods is to apply symbolic and/or numerical operations on the polynomials of the system to unveil the structure of the quotient algebra of the polynomial ring by the ideal, so that the root-solving problem can essentially be reduced to an eigenvalue problem; see e.g., [12] for a historical overview on the

*Submitted to the editors on January 19th, 2023.

Funding: This work was funded by (1) Flemish Government: (a) This research received funding from the Flemish Government (AI Research Program). Lieven De Lathauwer, Nithin Govindarajan and Raphaël Widdershoven are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. (b) This work was supported by the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no G0F6718N (SeLMA) (2) KU Leuven Internal Funds: iBOF/23/064, C14/22/096, C16/15/059 and IDN/19/014.

†Electrical Eng. (ESAT), KU Leuven, Leuven, Belgium (nithin.govindarajan@kuleuven.be, raphael.widdershoven@kuleuven.be, lieven.delathauwer@kuleuven.be)

‡Electrical and Computer Eng., University of California Santa Barbara, CA, United States (shiv@ucsb.edu)

method. There exists several means of accomplishing this reduction. The classical approach is to use Gröbner bases [9] or resultants [19] to construct the normal forms in the multiplication maps [14, Chapters 2 and 3]. The instability of these classical approaches has led to the development of border basis methods [38], and more recently, truncated normal forms [41, 51].

A fundamental object that arises frequently in these algebraic methods is the so-called Macaulay matrix[1], which generalizes the Sylvester resultant matrix of two univariate polynomials to the multivariate case [34]. To construct multiplication maps, particularly the null spaces of these matrices are of primary interest, since they have a direct correspondence with the quotient ring generated by the ideal. Along with the shift-invariance properties in the null space, this observation has allowed the authors in [3, 16, 17] to reformulate root-solving problem into a generalized eigenvalue problem starting from a numerical basis for the Macaulay null space. In [52], this generalized eigenvalue (GEVD) problem was further reformulated as a joint generalized eigenvalue (joint-GEVD) problem [21], or equivalently, a canonical polyadic decomposition (CPD) computation of a third-order tensor, by taking advantage of the commuting property of the multiplication maps. The algorithms in [5, 41, 51] also have as starting point a null space computation of a Macaulay-type matrix.

Irrespective of how the null space is further utilized, a major computational challenge shared by all aforementioned algorithms is the extraordinary dimensions of Macaulay-type matrices for even moderately-sized problems, making the null space basis computation prohibitively expensive. Classically, the algebraic geometry community has dealt with this challenge by exploiting possible sparsity structures that may be present in the equations, which allows for the construction of smaller resultant matrices [20]; see also the recent strides made in [5]. Nevertheless, Macaulay-type matrices are highly structured (even for the generic case), and limited investigation has taken place on how to exploit these structures directly in linear algebra computations [4, 41]. In particular, Macaulay-type matrices contain convolution operations, resulting in (quasi-)Toeplitz structures. Since these are matrices of low displacement rank [31], the question arises whether the tools of fast linear algebra for dense-structured matrices (see e.g., [10, 15, 30, 57]) can be utilized to design asymptotically faster algorithms.

1.1. Problem statement. In this paper, we confirm that asymptotically faster algorithms may indeed be formulated, at least satisfactorily for the bivariate case where the goal is to find all projective roots of the homogenized system. More specifically, we consider the (possibly) overdetermined set of equations

$$\Sigma : \begin{cases} p_1(x,y) &= \displaystyle\sum_{i=0}^{d_\Sigma} \sum_{j=0}^{d_\Sigma - i} c_{1ij} x^i y^j = 0 \\ & \vdots \\ p_S(x,y) &= \displaystyle\sum_{i=0}^{d_\Sigma} \sum_{j=0}^{d_\Sigma - i} c_{Sij} x^i y^j = 0 \end{cases}, \tag{1.1}$$

[1] In fact, many of the algebraic operations performed in these methods, including Gröbner basis constructions, can directly be related to linear algebra operations on this matrix itself; see e.g., [18, Section 3].

where it is assumed that for all $s = 1, \dots, S$, $p_s \in \mathbb{C}[x, y]$ is a polynomial of total degree[2] d_Σ, i.e., $c_{si(d_\Sigma - i)} \neq 0$ for some $i = 0, 1, \dots, d_\Sigma$. For $S \geqslant 2$, the system (1.1) is expected to admit d_Σ^2 (near) solutions (including multiplicities and so-called roots at infinity) if the set of equations are (approximately) consistent. These solutions are embedded in a d_Σ^2-dimensional null space of the Macaulay matrix $\mathrm{M}(d) \in \mathbb{C}^{m(d) \times n(d)}$ with $m(d), n(d) \sim d^2$ and $d \sim d_\Sigma$. Subsequently, with *current state-of-the-art techniques* (such as SVD or column-pivoted QR-decomposition), the cost of computing the null space will be $\mathcal{O}(d_\Sigma^6)$ floating point operations.

1.2. Contributions. The main contribution of this paper is to show that a numerical basis for the null space of the Macaulay matrix can be computed in $\mathcal{O}(d_\Sigma^5)$ floating point operations. To arrive to this result, we introduce a specialized algorithm that takes advantage of the "almost" upper-triangular Toeplitz block-(block-)Toeplitz structure of the Macaulay matrix in a non-graded lexicographic ordering of its entries (see Subsection 2.1). By applying displacement rank theory, it is shown that such matrices are efficiently converted into Cauchy-like matrices using Fast Fourier Transformations (FFTs) [29]. By adapting Ming Gu's variant of the Schur algorithm with approximate total pivoting [26], we then show that a compact representation of the right null space can be obtained for the Cauchy-transformed Macaulay matrix from a rank-revealing LU-factorization [37, 45]. Through inverse transformations, this representation can be converted to a numerical null space basis for the original matrix itself.

Central to the fast algorithm is the observation that the Macaulay matrix is of relatively *low* displacement rank, allowing for the Gauss steps in the Schur algorithm to be done quite efficiently. Technical contributions in this context are certain design choices in the algorithm to ensure stability, without sacrificing on (asymptotic) complexity. This includes some important implementation details on the re-orthonormalization updating strategy required for pivot selection, and a greedy heuristic to select near optimal parameters for the Cauchy conversion step. The performance of the algorithm is validated experimentally.

In addition to our main contribution above, we also show, but not implement, that the fast algorithm can be generalized for polynomial systems expressed in the Chebyshev basis; a problem of significant numerical importance [42, 43]. For this purpose, we describe a Chebyshev variant of the Macaulay matrix and reformulate the root-solving problem as a joint-GEVD problem in this setting as well. Although root-solving in the Chebyshev basis has already been studied in [41] within the context of truncated normal forms, our derivation of the joint-GEVD problem is new and insightful as it highlights the underlying Toeplitz-plus-Hankel structure of the Chebyshev-Macaulay matrix (see Subsection 5.1.1).

1.3. Related work. Structured matrices in the context of multivariate polynomial systems have been studied before in [39, 40] to design asymptotically faster algorithms through randomized techniques. The use of displacement rank theory in root-solving problems is also not entirely new. For instance, in [6, 7], it was observed how the Schur algorithm may be utilized to accelerate computations with Sylvester and Bézout matrices. Furthermore, [36] presented a modified version of Schur algorithm that determines the null space of a Toeplitz-like matrix, although motivated

[2]The proposed techniques introduced in this paper easily generalize to systems involving polynomials of varying degree, but for clarity of exposition, it is assumed that the degrees of all the polynomials in Σ are equal.

from a problem in control. Furthermore, the method differs fundamentally from ours as it is based-off a QR-decomposition and does not involve a Cauchy conversion.

1.4. Outline. The subsequent sections of this paper are organized as follows. Section 2 introduces the Macaulay matrix and the procedure of reducing the root-solving problem to a joint-GEVD problem. Section 3 discusses the fast algorithm for determining the null space of the Macaulay matrix. Section 4 presents some numerical experiments. Section 5 discusses how the fast algorithm can be generalized. We describe (i) a generalization of the algorithm for polynomial systems in the Chebyshev basis, and (ii) the law of diminishing returns when generalizing the algorithm to polynomial systems with more than two variables. Section 6 presents the conclusions.

Notation. Let $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{C}$ denote the set of integers, real and complex numbers. The imaginary number is denoted with ι, i.e., $\iota^2 = -1$. The projective complex plane, defined as set of points $(0,0,0) \neq (t,x,y) \in \mathbb{C}^3$ with $(t,x,y) \equiv (\lambda t, \lambda x, \lambda y)$ for any $0 \neq \lambda \in \mathbb{C}$, is denoted by $\mathbb{P}^2(\mathbb{C})$. $\mathbb{P}(\mathbb{C})$, on the other hand, denotes the projective complex line. The ring of polynomials over the complex field with indeterminates x and y, or indeterminates x, y and t is denoted respectively by $\mathbb{C}[x,y]$ and $\mathbb{C}[t,x,y]$. At times, where we would like to emphasize polynomial multiplication, the dot notation is adopted to express the product of two polynomials, e.g., $h \cdot p \in \mathbb{C}[x,y]$. The ideal generated by two polynomials $p, q \in \mathbb{C}[x,y]$ is expressed as $\mathcal{I}(p,q)$.

Capital Greek and Roman letters shall be used to denote matrices, while vectors are denoted with bold-faced characters. At our convenience, we use "Matlab" subscript notation to denote sub-blocks of vectors and matrices, e.g., $\mathrm{A}_{1:k,1}$ refers to the first k entries of the first column of the matrix A, while $\boldsymbol{v}_{(k+1):n}$ refers to the last $n-k$ entries of the vector $\boldsymbol{v} \in \mathbb{C}^n$. Certain commonly occurring families of vectors and matrices are denoted with special symbols. A vector of all zeros (ones) is denoted by $\mathbb{0}_n \in \mathbb{R}^n$ ($\mathbb{1}_n \in \mathbb{R}^n$), while a matrix of zeros (ones) is denoted by $\mathbb{0}_{m\times n} \in \mathbb{R}^{m\times n}$ ($\mathbb{1}_{m\times n} \in \mathbb{R}^{m\times n}$). The k-th unit vector of length n, with a one on the k-th position and zeros elsewhere, is denoted by $\boldsymbol{e}_{k,n} \in \mathbb{R}^n$. The n-by-n identity matrix is denoted by I_n, whereas $\mathrm{I}_{m,n}$ describes the m-by-n matrix with ones on the main diagonal and zeros elsewhere. Furthermore, for convenience we define

$$\mathrm{diag}(\boldsymbol{v}) := \begin{bmatrix} v_1 & & \\ & \ddots & \\ & & v_n \end{bmatrix}, \qquad \mathrm{diag}\{\mathrm{A}_i\}_{i=1}^n := \begin{bmatrix} \mathrm{A}_1 & & \\ & \ddots & \\ & & \mathrm{A}_n \end{bmatrix}.$$

At times, we may also use descending indices, e.g., $\mathrm{diag}\{\mathrm{A}_i\}_{i=n}^1 \equiv \mathrm{diag}\{\mathrm{A}_{n-i+1}\}_{i=1}^n$. The Kronecker product between two matrices is demarked with the symbol $\otimes$, i.e., for matrices $\mathrm{A} \in \mathbb{C}^{m\times n}$ and $\mathrm{B} \in \mathbb{C}^{p\times q}$,

$$\mathrm{A} \otimes \mathrm{B} := \begin{bmatrix} a_{11}\mathrm{B} & \cdots & a_{1n}\mathrm{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathrm{B} & \cdots & a_{mn}\mathrm{B} \end{bmatrix} \in \mathbb{C}^{pm\times qn}.$$

Let $\|\boldsymbol{v}\|_p := (\sum_{i=1}^n |v_i|^p)^{1/p}$, $\|\mathrm{A}\|_p := \max_{v\neq\mathbb{0}_n} \|\mathrm{A}\boldsymbol{v}\|_p / \|\boldsymbol{v}\|_p$, and $\|\mathrm{A}\|_\mathrm{F} := \sqrt{\sum_{i,j} |a_{ij}|^2}$. The rank of a matrix $\mathrm{A} \in \mathbb{C}^{m\times n}$ is denoted with $\mathrm{rank}\,\mathrm{A}$. The column and null spaces of A are denoted with $\mathrm{col}\ \mathrm{A}$ and $\mathrm{null}\,\mathrm{A}$, respectively. The symbols $(\cdot)^\top$ and $(\cdot)^*$, are used to denote transpose and conjugate transpose.

2. Macaulay-based method for polynomial root-solving. In this section, we review the Macaulay-based method for finding all projective common roots of (1.1). In Subsection 2.1 we introduce the Macaulay matrix, in Subsection 2.2 we summarize the properties of its null space viz-a-viz its relationship with the roots of the system, and in Subsection 2.3 we discuss how, starting from a null space basis, the root-solving problem is reduced to an eigenvalue problem or generalizations thereof [3, 16, 17, 52].

2.1. Macaulay matrix. Denote $\Delta d = d - d_\Sigma$ and define

$$m(d) := \tfrac{S}{2}(\Delta d + 1)(\Delta d + 2), \qquad n(d) := \tfrac{1}{2}(d + 1)(d + 2). \tag{2.1}$$

The Macaulay matrix $\mathrm{M}(d) \in \mathbb{C}^{m(d)\times n(d)}$ of degree $d \geqslant d_\Sigma$ is the matrix constructed from the polynomial coefficients in (1.1) such that its rows span the set of polynomials

$$\mathscr{M}(d) := \left\{ \sum_{s=1}^{S} h_s \cdot p_s : \quad h_s \in \mathbb{C}[x, y],\ \deg(h_s) = \Delta d \right\}. \tag{2.2}$$

The row and column indexing[3] used to describe this vector space is of course at our discretion. In this work, we adopt a non-graded lexicographic indexing (with $x < y$) as it reveals a (multi-level) Toeplitz structure that will be exploited in the method presented in Section 3. In other words, $x^{i_1}y^{j_1} < x^{i_2}y^{j_2}$ if $j_1 < j_2$, and in case $j_1 = j_2$, $i_1 < i_2$. The monomial terms x^iy^j with $i, j \leqslant d$ are ordered as

$$1, x, \ldots, x^d; y, xy, \ldots, x^dy; y^2, xy^2, \ldots, x^dy^2; ...; y^d, xy^d, \ldots, x^dy^d$$

but then excluding those terms that are not part of the collection $\left\{x^iy^j\right\}_{i,j\geqslant 0,\, i+j\leqslant d}$. The rows of $\mathrm{M}(d)$, which describe the set of "shifted" polynomials

$$\left\{x^iy^j \cdot p_1, \ldots, x^iy^j \cdot p_S\right\}_{i,j\geqslant 0,\, i+j\leqslant \Delta d},$$

are ordered in analogous manner, leading to indexing illustrated graphically in Figure 1.

As such, the entries of the Macaulay matrix may be described as follows. Recall that c_{skl} is the coefficient of polynomial $p_s \in \Sigma$ associated with the monomial term x^ky^l. For convenience, let $\boldsymbol{c}_{kl} := \begin{bmatrix} c_{1kl} & \cdots & c_{Skl}\end{bmatrix}^\top$ for $k \leqslant d_\Sigma - l$ and $\boldsymbol{c}_{kl} = \mathbb{0}_S$, otherwise. For $i = 0, 1, \ldots, d_\Sigma$ and $j = 0, 1, \ldots, \Delta d$, define the matrix $\mathrm{M}_{i,j} \in \mathbb{C}^{S(\Delta d+1-j)\times(d+1-i-j)}$ as

$$\mathrm{M}_{i,j} := \begin{bmatrix} \boldsymbol{c}_{0i} & \boldsymbol{c}_{1i} & \cdots & \boldsymbol{c}_{(d_\Sigma-i)i} & & & \\ & \boldsymbol{c}_{0i} & \boldsymbol{c}_{1i} & \cdots & \boldsymbol{c}_{(d_\Sigma-i)i} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \boldsymbol{c}_{0i} & \boldsymbol{c}_{1i} & \cdots & \boldsymbol{c}_{(d_\Sigma-i)i} \end{bmatrix}, \tag{2.3}$$

which represents the coefficients of the monomials with y^i repeated and shifted $\Delta d + 1 - j$ times. Then, for $d \geqslant d_\Sigma$, the Macaulay matrix associated with the polynomial system (1.1) is given by

$$\mathrm{M}(d) := \begin{bmatrix} \mathrm{M}_{0,0} & \mathrm{M}_{1,0} & \cdots & \mathrm{M}_{d_\Sigma,0} & & & \\ & \mathrm{M}_{0,1} & \mathrm{M}_{1,1} & \cdots & \mathrm{M}_{d_\Sigma,1} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \mathrm{M}_{0,\Delta d} & \mathrm{M}_{1,\Delta d} & \cdots & \mathrm{M}_{d_\Sigma,\Delta d} \end{bmatrix} \in \mathbb{C}^{m(d)\times n(d)}. \tag{2.4}$$

[3] Or for that matter, even the chosen polynomial basis. In Section 5, we describe how our ideas are extended to polynomial systems described in the Chebyshev basis.

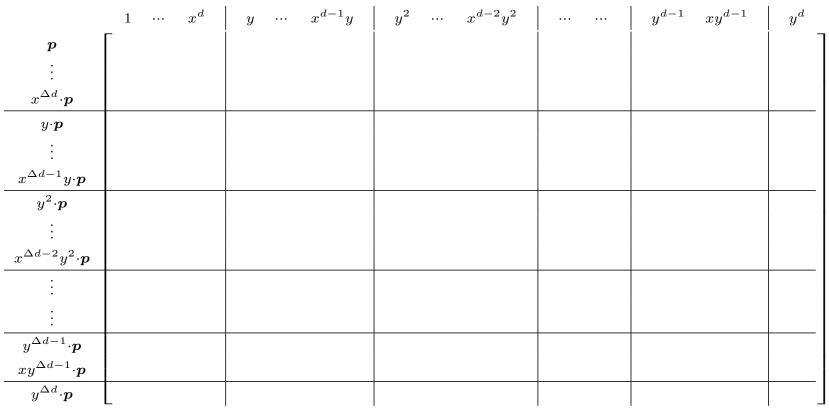


Fig. 1: The corresponding non-graded lexicographic indexing of the Macaulay matrix defined in (2.4). Here, $\boldsymbol{p} = (p_1, p_2, \ldots, p_S) \in (\mathbb{C}[x, y])^S$ and subsequently $x^i y^j \cdot \boldsymbol{p}$ is a shorthand for describing the polynomials $(x^i y^j \cdot p_1, x^i y^j \cdot p_2, \ldots, x^i y^j \cdot p_S)$.

To illustrate (2.4) with an example, consider the polynomial system

$$\Sigma : \quad \begin{cases} p_1(x,y) &= 1 + 6x + 4x^2 + 2y + 5xy + 3y^2 = 0 \\ p_2(x,y) &= 9 + 1x + 3x^2 + 8y + 7xy + 2y^2 = 0 \end{cases} .$$

The Macaulay matrix for $d = 4$ takes on the form

	1	x	x^2	x^3	x^4	y	xy	x^2y	x^3y	y^2	xy^2	x^2y^2	y^3	xy^3	y^4
p_1	1	6	4			2	5			3					
p_2	9	1	3			8	7			2					
xp_1		1	6	4			2	5			3				
xp_2		9	1	3			8	7			2				
x^2p_1			1	6	4			2	5			3			
x^2p_2			9	1	3			8	7			2			
yp_1						1	6	4		2	5		3		
yp_2						9	1	3		8	7		2		
xyp_1							1	6	4		2	5		3	
xyp_2							9	1	3		8	7		2	
y^2p_1										1	6	4	2	5	3
y^2p_2										9	1	3	8	7	2

.

The Macaulay matrix (2.4), for the chosen ordering, has an upper-triangular Toeplitz block-(block-)Toeplitz matrix[4], but then with rows corresponding with polynomial shifts of degree greater than Δd and columns corresponding with monomial terms of degree greater than d removed accordingly. That is, we may write

$$\mathrm{M}(d) := \operatorname{diag}\left\{\mathrm{I}_{i,\Delta d+1} \otimes \mathrm{I}_S\right\}_{i=\Delta d+1}^{1} \mathrm{M}^{\mathrm{tpz}}(d) \operatorname{diag}\left\{\mathrm{I}_{d+1,j}\right\}_{j=d+1}^{1} \tag{2.5}$$

[4]An upper-triangular block Toeplitz matrix, where each block element is again upper-triangular (block-)Toeplitz.

where

$$\mathrm{M}^{\mathrm{tpz}}(d) := \begin{bmatrix} \mathrm{M}_0^{\mathrm{tpz}} & \mathrm{M}_1^{\mathrm{tpz}} & \cdots & \mathrm{M}_{d_\Sigma}^{\mathrm{tpz}} & & & \\ & \mathrm{M}_0^{\mathrm{tpz}} & \mathrm{M}_1^{\mathrm{tpz}} & \cdots & \mathrm{M}_{d_\Sigma}^{\mathrm{tpz}} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \mathrm{M}_0^{\mathrm{tpz}} & \mathrm{M}_1^{\mathrm{tpz}} & \cdots & \mathrm{M}_{d_\Sigma}^{\mathrm{tpz}} \end{bmatrix} \in \mathbb{C}^{S(\Delta d+1)^2 \times (d+1)^2},$$

$$\mathrm{M}_j^{\mathrm{tpz}} := \begin{bmatrix} \boldsymbol{c}_{0j} & \boldsymbol{c}_{1j} & \cdots & \boldsymbol{c}_{d_\Sigma j} & & & \\ & \boldsymbol{c}_{0j} & \boldsymbol{c}_{1j} & \cdots & \boldsymbol{c}_{d_\Sigma j} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \boldsymbol{c}_{0j} & \boldsymbol{c}_{1j} & \cdots & \boldsymbol{c}_{d_\Sigma j} \end{bmatrix} \in \mathbb{C}^{S(\Delta d+1) \times (d+1)},$$

for $j = 0, 1, \ldots, d_\Sigma$.

2.2. Properties of the Macaulay null space. For $S \geqslant 2$, the Macaulay matrix eventually grows into a tall matrix with more rows than columns for sufficiently large values of d. The matrix is however rank deficient and has a nontrivial right null space.

The right null space of the Macaulay matrix (2.4) is closely linked to the set of common roots of the system (1.1), or more specifically, its homogenization

$$(2.6)\qquad \Sigma_h : \begin{cases} p_{1,h}(t,x,y) & := & t^{d_\Sigma} \cdot p_1(x/t, y/t) = 0 \\ & \vdots & \\ p_{S,h}(t,x,y) & := & t^{d_\Sigma} \cdot p_S(x/t, y/t) = 0 \end{cases}$$

in the projective complex plane $\mathbb{P}^2(\mathbb{C})$. Indeed, if $\mathbb{v}_d(t,x,y) \in \mathbb{C}^{n(d)}$ defines the vector

$$(2.7)\qquad \mathbb{v}_d(t,x,y) := t^d \cdot \mathbb{v}_{d,x,y}(x/t, y/t),$$

where

$$\mathbb{v}_{d,x,y}(x,y) := \begin{bmatrix} \mathbb{v}_{d,x}^\top(x) & y \cdot \mathbb{v}_{d-1,x}^\top(x) & \cdots & y^d \cdot \mathbb{v}_{0,x}^\top(x) \end{bmatrix}^\top \in \mathbb{C}^{n(d)},$$

$$\mathbb{v}_{d,x}(x) := \begin{bmatrix} 1 & x & \cdots & x^d \end{bmatrix}^\top \in \mathbb{C}^{d+1},$$

we observe that for every common root $(t^*, x^*, y^*) \in \mathbb{P}^2(\mathbb{C})$ of Σ_h, it must hold that $\mathbb{v}_d(t^*, x^*, y^*) \in \operatorname{null} \mathrm{M}(d)$. In relation to the original system Σ, we may place the roots of Σ_h in two distinct categories: if $t \neq 0$, $(t^*, x^*, y^*) \in \mathbb{P}^2(\mathbb{C})$ is considered to be an *affine root* of Σ_h, otherwise it is called a *root at infinity*. Affine roots of Σ_h have a direct correspondance with the roots of the original system Σ in affine space. That is, since $(t^*, x^*, y^*) \equiv (1, x^*/t^*, y^*/t^*)$ in $\mathbb{P}^2(\mathbb{C})$, the point $(x^*/t^*, y^*/t^*) \in \mathbb{C}^2$ will be a root of Σ because of the identity $p_{s,h}(1, x/t, y/t) = p_s(x/t, y/t)$. Roots at infinity, on the other hand, do not relate to any roots of Σ. Instead, they are roots of the homogeneous system

$$\Sigma_\infty : \begin{cases} p_{1,\infty}(x,y) & := & p_{1,h}(0,x,y) = \displaystyle\sum_{i=0}^{d_\Sigma} c_{1i(d_\Sigma - i)} x^i y^{d_\Sigma - i} = 0 \\ & \vdots & \\ p_{S,\infty}(x,y) & := & p_{S,h}(0,x,y) = \displaystyle\sum_{i=0}^{d_\Sigma} c_{Si(d_\Sigma - i)} x^i y^{d_\Sigma - i} = 0 \end{cases}$$

in $\mathbb{P}(\mathbb{C})$. From the fundamental theorem of algebra, it can be shown that a root at infinity only occurs if all homogeneous polynomials in Σ_∞ share a nontrivial common factor. Mathematically, the possibility of this occurring for a generic system is zero, yet it should be noted that in many structured polynomial systems which arise in practice, this property no longer holds true; see e.g., [46, 50] for examples.

Nevertheless, here we focus on the generic case with an interest in finding *all* roots of the homogenized system (2.6). If the polynomials in Σ_h do not share any common nontrivial factors, i.e., $p_{s,h} \neq f \cdot g_{s,h}$ for some non-constant polynomial $f \in \mathbb{C}[t, x, y]$, this number will turn out to be finite. Specifically, if $S = 2$, Bézout's theorem (see e.g. [28, Theorem 7.7]) applies and the number of roots, accounting for multiplicity, equals d_Σ^2. On the other hand, for overdetermined systems, the number of roots will generically be zero[5]. To still provide a proper complexity analysis later in Subsection 3.3, we shall assume that two coprime polynomials in Σ generate the entire ideal formed by all polynomials of the system so that we obtain a consistent set of equations. That is,

$$\exists p, q \in \Sigma, \text{ with } p \text{ and } q \text{ coprime, such that } \mathcal{I}(p, q) = \mathcal{I}(\Sigma)\ . \tag{2.8}$$

In such a circumstance, the homogenized system Σ_h will again have d_Σ^2 common roots. From a practical standpoint, it is sensible to assume condition (2.8) since it idealizes a scenario of an overdetermined system being ϵ-close to a square system, i.e., where condition (2.8) is only satisfied in an approximate sense.

2.3. Recovering the roots from the Macaulay null space. As pointed out in the introduction, there exist numerous ways to reformulate the root-solving problem into an eigenvalue problem. In this section, we review the method in [52], which builds upon the foundational work in [3, 16, 17]. In this approach, the root-solving problem is reduced to a joint generalized eigenvalue (joint-GEVD) problem, or equivalently a CPD computation. For simplicity of exposition, we shall assume for the remainder of this section that all roots of Σ_h are simple, i.e., the multiplicities equal one[6]. Note however that this assumption can be removed and properly addressed through, for instance, the frameworks presented in [11] or [53].

Let $\{(t_i, x_i, y_i) \in \mathbb{P}^2(\mathbb{C})\}_{i=1}^{d_\Sigma^2}$ denote the set of common roots of Σ_h and define the multivariate Vandermonde matrix as

$$\mathbb{V}(d) = \begin{bmatrix} \mathbb{v}_d(t_1, x_1, y_1) & \cdots & \mathbb{v}_d(t_{d_\Sigma^2}, x_{d_\Sigma^2}, y_{d_\Sigma^2}) \end{bmatrix} \in \mathbb{C}^{n(d) \times d_\Sigma^2}. \tag{2.9}$$

It is clear that $\operatorname{col} \mathbb{V}(d) \subseteq \operatorname{null} \mathrm{M}(d)$. It turns out that this containment can be strengthened to an equality. In fact, there exists a so-called degree of regularity d^* for which the nullity of Macaulay matrix stabilizes to the number of roots in the system, which in the case of (2.6) with condition (2.8) implies that $\dim \operatorname{null} \mathrm{M}(d) = d_\Sigma^2$ for all $d \geqslant d^*$. Subsequently,

$$r(d) := \operatorname{rank} \mathrm{M}(d) = n(d) - d_\Sigma^2, \qquad d \geqslant d^*. \tag{2.10}$$

Upper bounds on the degree of regularity relate back to original work by F.S. Macaulay [34] and can be found, for example, in [14, Section 3.4]. Specifically, the degree of

[5] This can be interpreted as a generalization of the statement that an overdetermined linear system typically has no exact solution.

[6] The multiplicity quantifies intuitively in how many distinct intersections a common root of two plane curves (described by the vanishing set of the respective polynomials) disperses under arbitrary small perturbation. For generic intersections, this number equals one.

regularity for the bivariate system (1.1) is bounded by

$$d^* \leqslant 2d_\Sigma - 2. \tag{2.11}$$

The degree of regularity is often attained well before the bound in (2.11). In practice, one uses recursive approaches to construct the null space to avoid forming unnecessarily large Macaulay matrices [4,41]. In our analysis later in Subsection 3.3, we shall nonetheless use (2.11) to provide upper bounds on the complexity.

The reduction of the roots solving problem to a joint-GEVD problem [21] takes advantage of the fact that the columns of (2.9) form a basis for $\operatorname{null} \mathrm{M}(d)$ if $d \geqslant d^*$. In particular, one exploits the shift-invariant structure in (2.9) as follows. Let $\mathrm{S}_t(d), \mathrm{S}_x(d), \mathrm{S}_y(d) \in \mathbb{R}^{n(d-1)\times n(d)}$ denote the shift-matrices

$$\mathrm{S}_t(d) = \operatorname{diag}\{\mathrm{S}_{t,d-i}\}_{i=0}^{d}, \quad \mathrm{S}_{t,i} = \begin{bmatrix} 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{i\times(i+1)},$$

$$\mathrm{S}_x(d) = \operatorname{diag}\{\mathrm{S}_{x,d-i}\}_{i=0}^{d}, \quad \mathrm{S}_{x,i} = \begin{bmatrix} 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{bmatrix} \in \mathbb{R}^{i\times(i+1)},$$

and

$$\mathrm{S}_y(d) = \begin{bmatrix} \mathbb{0}_{d\times(d+1)} & \mathrm{I}_d & & & & \\ & \mathbb{0}_{(d-1)\times d} & \mathrm{I}_{d-1} & & & \\ & & \mathbb{0}_{(d-2)\times(d-1)} & \ddots & & \\ & & & \ddots & \mathrm{I}_2 & \\ & & & & \mathbb{0}_{1\times 2} & 1 \end{bmatrix}.$$

Since $\mathrm{S}_h(d+1)\mathbb{v}_{d+1}(t,x,y) = h \cdot \mathbb{v}_d(t,x,y)$ for $h = \{t,x,y\}$, we obtain the relations

$$\mathrm{S}_t(d+1)\mathbb{V}(d+1) = \mathbb{V}(d)\mathrm{D}_t, \tag{2.12a}$$

$$\mathrm{S}_x(d+1)\mathbb{V}(d+1) = \mathbb{V}(d)\mathrm{D}_x, \tag{2.12b}$$

$$\mathrm{S}_y(d+1)\mathbb{V}(d+1) = \mathbb{V}(d)\mathrm{D}_y, \tag{2.12c}$$

where

$$\mathrm{D}_t = \operatorname{diag}(t_1,\dots,t_{d_\Sigma^2}), \quad \mathrm{D}_x = \operatorname{diag}(x_1,\dots,x_{d_\Sigma^2}), \quad \mathrm{D}_y = \operatorname{diag}(y_1,\dots,y_{d_\Sigma^2}). \tag{2.13}$$

Suppose that the columns of $\mathrm{N}(d)$ are a basis for $\operatorname{null} \mathrm{M}(d)$. Since the columns of $\mathrm{N}(d) \in \mathbb{C}^{n(d)\times d_\Sigma^2}$ span the same subspace as the columns of $\mathbb{V}(d)$ for $d \geqslant d^*$, there exists an invertible matrix $\mathrm{A} \in \mathbb{C}^{d_\Sigma^2\times d_\Sigma^2}$ such that $\mathrm{N}(d)\mathrm{A} = \mathbb{V}(d)$. Substitution of this identity into (2.12) yields a joint-GEVD problem. That is, given the matrices

$$\mathrm{G}_1 := \mathrm{S}_t(d+1)\mathrm{N}(d+1), \quad \mathrm{G}_2 := \mathrm{S}_x(d+1)\mathrm{N}(d+1), \quad \mathrm{G}_3 := \mathrm{S}_y(d+1)\mathrm{N}(d+1),$$

find an A that simultaneously diagonalizes $\mathrm{G}_i \in \mathbb{C}^{n(d)\times d_\Sigma^2}$, i.e.,

$$\mathrm{G}_1\mathrm{A} = \mathbb{V}(d)\mathrm{D}_t, \quad \mathrm{G}_2\mathrm{A} = \mathbb{V}(d)\mathrm{D}_x, \quad \mathrm{G}_3\mathrm{A} = \mathbb{V}(d)\mathrm{D}_y. \tag{2.14}$$

The set of matrix equations (2.14) can be rephrased as the CPD of a tensor whose frontal slices are given by G_i for $i = 1,2,3$. Well-established reliable numerical methods exist to compute CPDs of tensors; see e.g., [21,47,56], and the references therein. A schematic summary of the entire method is shown in Figure 2.

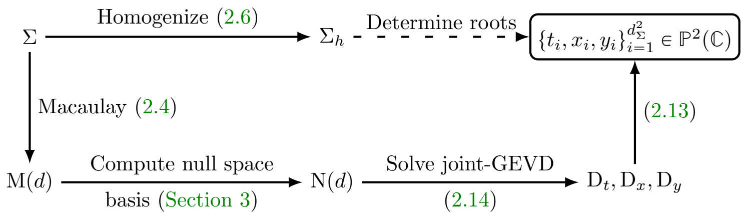


Fig. 2: A schematic overview of how all projective roots of the (homogenized) system are found. Our objective is to determine the roots of the homogenized system (dashed line). This is achieved by following the steps given by the solid lines, i.e., first computing a basis for the null space of the Macaulay matrix, and then solving the joint-GEVD problem Equation (2.14).

3. Fast determination of the Macaulay null space. The Macaulay matrix (2.4) has an almost Toeplitz-block-(block-Toeplitz) structure as described in detail in Subsection 2.1. We describe an efficient method to determine a numerical basis for the right null space of such a matrix. The method proceeds in three steps:

1. Apply unitary transformations $\Phi \in \mathbb{C}^{m(d)\times m(d)}$ and $\Psi \in \mathbb{C}^{n(d)\times n(d)}$ such that $\Phi \mathrm{M}(d) \Psi =: \hat{\mathrm{M}}(d)$ attains the structure of a Cauchy-like matrix.
2. Compute a fast rank-revealing LU-factorization of $\hat{\mathrm{M}}(d)$ using the Schur algorithm to obtain a basis for its null space $\hat{\mathrm{N}}(d)$.
3. Recover the null space of the original Macaulay matrix from $\mathrm{N}(d) = \Psi\hat{\mathrm{N}}(d)$.

A schematic outline of the method is presented in Figure 3. Referring to this outline, Subsection 3.1 provides the details of how the Macaulay matrix is efficiently converted into a Cauchy-like matrix. Subsection 3.2 discusses the details of finding an efficient null space representation for this Cauchy-transformed Macaulay matrix using the Schur algorithm. The recovery of the null space for the actual Macaulay matrix becomes a trivial step, since an expression for Ψ has already been derived in Subsection 3.1. In Subsection 3.3, a summary of the algorithm is given along with an analysis of its asymptotic complexity.

Remark 3.1. Mind that our method will always produce a complex basis for the null space, even if all coefficients in (1.1) are real. If a real basis is so specifically desired in an application, one may obtain this by working with a displacement equation of the type in (5.8), instead of the displacement equation in (3.2) that will be presented shortly.

3.1. Fast conversion of Macaulay matrices into Cauchy-like matrices. This section details how one efficiently converts Macaulay matrices into Cauchy-like matrices. The described method relies on concepts from displacement rank theory; see [31] for a comprehensive review on the subject or [10] for a more concise introduction.

3.1.1. Low displacement-rank structure of Macaulay matrices. Let $\varphi \in \mathbb{C}$ be of unit modulus, i.e., $|\varphi| = 1$, and denote

$$\mathrm{Z}_{p,\varphi} = \begin{bmatrix} & & & \varphi \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix} \in \mathbb{C}^{p\times p}, \tag{3.1}$$

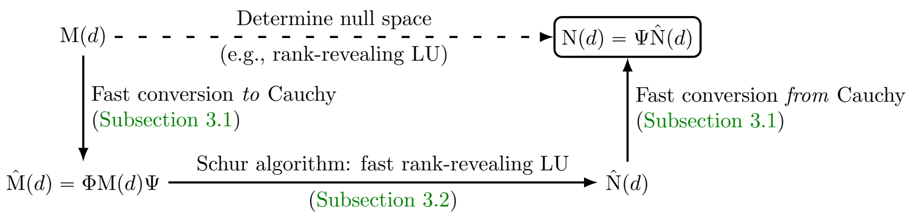


Fig. 3: A schematic outline of the fast algorithm. Our objective is to determine the null space of the Macaulay matrix (dashed line). This is achieved by following the steps given by the solid lines, i.e., first perform a Cauchy conversion, run the Schur algorithm to compute a rank-revealing LU-factorization, perform an inverse transformation to recover the null-space of the original matrix.

for $p \geqslant 2$, and $\mathrm{Z}_{1,\varphi} = \varphi$ in the special case when $p = 1$. Consider the displacement operator $\mathscr{D} : \mathbb{C}^{m(d)\times n(d)} \to \mathbb{C}^{m(d)\times n(d)}$ defined as the linear map

$$\mathscr{D}: \quad \mathrm{X} \mapsto \operatorname{diag}\left\{\mathrm{Z}_{i,1} \otimes \mathrm{I}_S\right\}_{i=\Delta d+1}^{1} \mathrm{X} - \mathrm{X} \operatorname{diag}\left\{\mathrm{Z}_{j,\varphi_j}\right\}_{j=d+1}^{1}, \tag{3.2}$$

where $\{\varphi_j\}_{j=1}^{d+1}$ are chosen particularly such that (3.2) remains bijective[7]. A practical choice for these parameters will be discussed in Subsection 4.1.1. For Macaulay matrices, the image under the displacement operator are matrices of (relatively) low rank. Indeed, applying (3.2) onto (2.4) yields

$$\mathscr{D}\left\{\mathrm{M}(d)\right\} = \breve{\mathrm{M}} = \begin{bmatrix} \breve{\mathrm{M}}_{0,0} & \breve{\mathrm{M}}_{1,0} & \cdots & \breve{\mathrm{M}}_{d_\Sigma,0} & & & \\ & \breve{\mathrm{M}}_{0,1} & \breve{\mathrm{M}}_{1,1} & \cdots & \breve{\mathrm{M}}_{d_\Sigma,1} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \breve{\mathrm{M}}_{0,\Delta d} & \breve{\mathrm{M}}_{1,\Delta d} & \cdots & \breve{\mathrm{M}}_{d_\Sigma,\Delta d} \end{bmatrix}, \tag{3.3}$$

where $\breve{\mathrm{M}}_{j-i,i} := \left(\mathrm{Z}_{\Delta d+1-i,1} \otimes \mathrm{I}_S\right) \mathrm{M}_{j-i,i} - \mathrm{M}_{j-i,i} \mathrm{Z}_{d+1-j,\varphi_{d+1-j}} \in \mathbb{C}^{S(\Delta d+1-i)\times(d+1-j)}$ are matrices of the form

$$\breve{\mathrm{M}}_{j-i,i} = \begin{bmatrix} \mathbb{0}_S & \cdots & \mathbb{0}_S & \boldsymbol{c}_{0(j-i)} & \cdots & \boldsymbol{c}_{(d_\Sigma-j+i-1)(j-i)} & \boldsymbol{c}_{(d_\Sigma-j+i)(j-i)} \\ \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S \end{bmatrix} - \begin{bmatrix} \boldsymbol{c}_{1(j-i)} & \cdots & \boldsymbol{c}_{(d_\Sigma-j+i)(j-i)} & \mathbb{0}_S & \cdots & \mathbb{0}_S & \varphi_{d+1-j}\boldsymbol{c}_{0(j-i)} \\ \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S & \cdots & \mathbb{0}_S & \mathbb{0}_S \end{bmatrix}. \tag{3.4}$$

Since $\operatorname{rank} \breve{\mathrm{M}}_{j-i,i} \leqslant S$, we may further deduce that

$$\operatorname{rank} \mathscr{D}\left\{\mathrm{M}(d)\right\} \leqslant S(\Delta d + 1) = S\,(d+1) - S d_\Sigma =: \rho(d). \tag{3.5}$$

[7] Let $\lambda(\mathrm{A}) \subset \mathbb{C}$ and $\lambda(\mathrm{B}) \subset \mathbb{C}$ denote the spectrum of $\mathrm{A} \in \mathbb{C}^{m\times m}$ and $\mathrm{B} \in \mathbb{C}^{n\times n}$, respectively. The linear operator $\mathscr{L} : X \mapsto AX - XB$ is invertible if, and only if, $\lambda(\mathrm{A}) \cap \lambda(\mathrm{B}) = \varnothing$.

This reveals that, while both the height and width of the Macaulay matrix grow *quadratically* with respect to d, the rank of the displaced Macaulay matrix grows only *linearly* with d. Specifically, when d equals the upper bound on the degree of regularity (2.11), the Macaulay matrix is an $\frac{S}{2}(d_\Sigma - 1)d_\Sigma$ by $\frac{1}{2}(2d_\Sigma - 1)d_\Sigma$ matrix, while its displacement has rank of at most $S(d_\Sigma - 1)$. This critical observation is what allows for a fast algorithm since it will substantially reduce the cost of performing Gaussian elimination (to be discussed in Subsection 3.2).

3.1.2. Cauchy representation of Macaulay matrices. Matrices of the kind in (2.4) are easily converted into Cauchy-like matrices through unitary transformations. That is, there exist unitary matrices $\Phi \in \mathbb{C}^{m(d)\times m(d)}$, $\Psi \in \mathbb{C}^{n(d)\times n(d)}$ such that $\hat{\mathrm{M}}(d) := \Phi \mathrm{M}(d)\Psi \in \mathbb{C}^{m(d)\times n(d)}$ is Cauchy-like and thus has entries of the form

$$\left[\hat{\mathrm{M}}(d)\right]_{ij} := \left[\Phi \mathrm{M}(d)\Psi\right]_{ij} = \frac{\boldsymbol{u}_i^* \boldsymbol{v}_j}{\mu_i - \nu_j}, \qquad \boldsymbol{u}_i, \boldsymbol{v}_j \in \mathbb{C}^{\rho(d)}. \tag{3.6}$$

To see how Φ and Ψ should be picked, observe at first that (3.6) satisfies the displacement equation

$$\hat{\mathscr{D}}\left\{\hat{\mathrm{M}}(d)\right\} := \operatorname{diag}(\boldsymbol{\mu})\hat{\mathrm{M}}(d) - \hat{\mathrm{M}}(d)\operatorname{diag}(\boldsymbol{\nu}) = \begin{bmatrix} \boldsymbol{u}_1^* \\ \vdots \\ \boldsymbol{u}_{m(d)}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_{n(d)} \end{bmatrix} \tag{3.7}$$

and hence, it is convenient at times to denote a Cauchy-like matrix just in terms of its "generators", i.e.,

$$\hat{\mathrm{M}}(d) = \mathscr{C}\left(\boldsymbol{\mu}, \boldsymbol{\nu}, \mathrm{U}, \mathrm{V}\right), \tag{3.8}$$

with $\mathrm{U} \in \mathbb{C}^{m(d)\times\rho(d)}$ and $\mathrm{V} \in \mathbb{C}^{n(d)\times\rho(d)}$ defined as

$$\mathrm{U} := \begin{bmatrix} \boldsymbol{u}_1^* \\ \vdots \\ \boldsymbol{u}_{m(d)}^* \end{bmatrix}, \qquad \mathrm{V} := \begin{bmatrix} \boldsymbol{v}_1^* \\ \vdots \\ \boldsymbol{v}_{n(d)}^* \end{bmatrix}.$$

The displacement equation in (3.2) can be molded into the displacement equation of (3.7) by substituting the eigen-decomposition of (3.1) into (3.2) and manipulating the expression. Indeed, let $\omega_p := \exp(-2\pi\imath/p)$ and observe that (3.1) decomposes into

$$\mathrm{Z}_{p,\varphi} = (\mathrm{D}_{p,\varphi} F_p)(\varphi^{1/p}\Omega_p)(\mathrm{D}_{p,\varphi}\mathrm{F}_p)^{-1},$$

where $\mathrm{D}_{p,\varphi} := \operatorname{diag}(1, \varphi^{-1/p}, \ldots, \varphi^{-(p-1)/p})$, $\Omega_p := \operatorname{diag}(1, \bar{\omega}_p, \ldots, \bar{\omega}_p^{p-1})$, and $\mathrm{F}_p \in \mathbb{C}^{p\times p}$ is the Discrete Fourier Transform (DFT) matrix, i.e., $[\mathrm{F}_p]_{ij} := \frac{1}{\sqrt{p}}\omega_p^{(i-1)(j-1)}$. By setting

$$\Phi := \operatorname{diag}\left\{\mathrm{F}_i^* \otimes \mathrm{I}_S\right\}_{i=\Delta d+1}^{1}, \qquad \Psi := \operatorname{diag}\left\{\mathrm{D}_{j,\varphi_j}\mathrm{F}_j\right\}_{j=d+1}^{1}, \tag{3.9}$$

$$\operatorname{diag}(\boldsymbol{\mu}) := \operatorname{diag}\left\{\Omega_i \otimes \mathrm{I}_S\right\}_{i=\Delta d+1}^{1}, \qquad \operatorname{diag}(\boldsymbol{\nu}) := \operatorname{diag}\left\{\varphi_j^{1/j}\Omega_j\right\}_{j=d+1}^{1}, \tag{3.10}$$

one can show from a sequence of algebraic manipulations that (3.6) satisfies the relation

$$\hat{\mathscr{D}}\left\{\hat{\mathrm{M}}(d)\right\} = \Phi\mathscr{D}\left\{\mathrm{M}(d)\right\}\Psi = \Phi\breve{\mathrm{M}}\Psi = \mathrm{U}\mathrm{V}^*. \tag{3.11}$$

3.1.3. Fast Cauchy conversion using FFTs. By (3.5) and (3.11), we have that

$$\operatorname{rank} \hat{\mathscr{D}} \left\{ \hat{\mathrm{M}}(d) \right\} = \operatorname{rank} \mathscr{D} \left\{ \mathrm{M}(d) \right\} \leqslant \rho(d),$$

and finding the representation (3.6) is equivalent to just finding a low-rank factorization $\mathrm{U}\mathrm{V}^*$ for $\Phi \mathscr{D}\{\mathrm{M}(d)\}\Psi$, as the denominator coefficients $\mu_i, \nu_j \in \mathbb{C}$ are already cast in stone by (3.10). A pair of matrices U and V can be determined rather efficiently. To see this, observe that by substitution of (3.9) into (3.3), we must apply the transformation

$$\check{\mathrm{M}}_{j-i,i} \mapsto \left(\mathrm{F}^*_{\Delta d+1-i} \otimes \mathrm{I}_S\right) \check{\mathrm{M}}_{j-i,i} \left(\mathrm{D}_{d+1-j,\varphi_{d+1-j}} \mathrm{F}_{d+1-j}\right) =: \mathrm{U}_i \mathrm{V}^*_{j-i,i}.$$

Since, by (3.4), $\check{M}_{j-i,i}$ factors into

$$\mathrm{I}_{S(\Delta d+1-i),S} \left(\begin{bmatrix} \mathbb{0}_S & \cdots & \mathbb{0}_S & \boldsymbol{c}_{0(j-i)} & \cdots & \boldsymbol{c}_{(d_\Sigma-j+i-1)(j-i)} & \boldsymbol{c}_{(d_\Sigma-j+i)(j-i)} \end{bmatrix} - \begin{bmatrix} \boldsymbol{c}_{1(j-i)} & \cdots & \boldsymbol{c}_{(d_\Sigma-j+i)(j-i)} & \mathbb{0}_S & \cdots & \mathbb{0}_S & \varphi_{d+1-j}\boldsymbol{c}_{0(j-i)} \end{bmatrix} \right),$$

we may write $\mathrm{U}_i \in \mathbb{C}^{S(\Delta d+1-i)\times S}$ and $\mathrm{V}_{j-i,i} \in \mathbb{C}^{(d+1-j)\times S}$ as

$$\mathrm{U}_i = \left(\mathrm{F}^*_{\Delta d+1-i} \otimes \mathrm{I}_S\right)\left(\boldsymbol{e}_{1,\Delta d+1-i} \otimes \mathrm{I}_S\right) = \frac{1}{\sqrt{\Delta d+1-i}} \left(\mathbb{1}_{\Delta d+1-i} \otimes \mathrm{I}_S\right),$$

$$\mathrm{V}_{j-i,i} = \mathrm{F}^*_{d+1-j}\mathrm{D}^*_{d+1-j,\varphi_j} \left(\begin{bmatrix} \mathbb{0}_{1\times S} \\ \vdots \\ \mathbb{0}_{1\times S} \\ \boldsymbol{c}^*_{0(j-i)} \\ \vdots \\ \boldsymbol{c}^*_{(d_\Sigma-j+i-1)(j-i)} \\ \boldsymbol{c}^*_{(d_\Sigma-j+i)(j-i)} \end{bmatrix} - \begin{bmatrix} \boldsymbol{c}^*_{1(j-i)} \\ \vdots \\ \boldsymbol{c}^*_{(d_\Sigma-j+i)(j-i)} \\ \mathbb{0}_{1\times S} \\ \vdots \\ \mathbb{0}_{1\times S} \\ \bar{\varphi}_{d+1-j}\boldsymbol{c}^*_{0(j-i)} \end{bmatrix} \right).$$

Subsequently,

$$\mathrm{U} = \operatorname{diag}\left\{\mathrm{U}_i\right\}_{i=0}^{\Delta d}, \qquad \mathrm{V} = \begin{bmatrix} \mathrm{V}_{0,0} & & \\ \vdots & \ddots & \\ \mathrm{V}_{d_\Sigma,0} & & \mathrm{V}_{0,\Delta d} \\ & \ddots & \vdots \\ & & \mathrm{V}_{d_\Sigma,\Delta d} \end{bmatrix}. \tag{3.12}$$

3.2. Fast null space computation of Cauchy-like matrices. This section details how one efficiently computes a numerical basis for the right null space of the Cauchy-like matrix (3.6) through a rank-revealing LU-factorization [37, 45].

3.2.1. Rank-revealing LU-factorizations. Assume that condition (2.8) is satisfied and that $d \geqslant d^*$ so that the Macaulay matrix has rank $r(d)$ as specified in (2.10). Following the definition in [37], in a rank-revealing LU-factorization of $\hat{\mathrm{M}}(d)$, the goal is to find row and column permutations $\Pi_1 \in \mathbb{R}^{m(d)\times m(d)}$ and $\Pi_2 \in \mathbb{R}^{n(d)\times n(d)}$ such that[8]

$$\Pi_1 \hat{\mathrm{M}}(d) \Pi_2 = \begin{bmatrix} \hat{\mathrm{M}}_{11} & \hat{\mathrm{M}}_{12} \\ \hat{\mathrm{M}}_{21} & \hat{\mathrm{M}}_{22} \end{bmatrix},$$

[8] Mind that $\hat{\mathrm{M}}_{ij}$ are sub-blocks of the permuted matrix $\Pi_1 \hat{\mathrm{M}}(d) \Pi_2$ and not of $\hat{\mathrm{M}}$ itself!

with partition blocks $\hat{\mathrm{M}}_{11} \in \mathbb{C}^{r(d)\times r(d)}$, $\hat{\mathrm{M}}_{12} \in \mathbb{C}^{r(d)\times d_\Sigma^2}$, $\hat{\mathrm{M}}_{21} \in \mathbb{C}^{(m(d)-r(d))\times r(d)}$, and $\hat{\mathrm{M}}_{22} \in \mathbb{C}^{(m(d)-r(d))\times d_\Sigma^2}$, factors into

$$\Pi_1 \hat{\mathrm{M}}(d) \Pi_2 = \begin{bmatrix} \mathrm{I}_{r(d)} & \\ \hat{\mathrm{M}}_{21}\hat{\mathrm{M}}_{11}^{-1} & \mathrm{I}_{d_\Sigma^2} \end{bmatrix} \begin{bmatrix} \hat{\mathrm{M}}_{11} & \\ & \hat{\mathrm{M}}_{22} - \hat{\mathrm{M}}_{21}\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12} \end{bmatrix} \begin{bmatrix} \mathrm{I}_{r(d)} & \hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12} \\ & \mathrm{I}_{d_\Sigma^2} \end{bmatrix},$$

where

$$\sigma_i(\hat{\mathrm{M}}_{11}) \geqslant \frac{\sigma_i(\hat{\mathrm{M}}(d))}{q(m,n,r)}, \qquad \sigma_j(\hat{\mathrm{M}}_{22} - \hat{\mathrm{M}}_{21}\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12}) \leqslant \sigma_{j+r(d)}(\hat{\mathrm{M}}(d)) q(m,n,r), \tag{3.13}$$

for $i = 1, \ldots, r(d)$, $j = 1, \ldots, d_\Sigma^2$, and $q(m,n,r)$ an expression that is a low degree polynomial in the matrix dimensions and rank. Since $\sigma_{r(d)}(\hat{\mathrm{M}}(d)) \gg \sigma_{r(d)+1}(\hat{\mathrm{M}}(d)) \approx 0$ in a numerical setting, the bounds (3.13) ensure that the Schur complement $\hat{\mathrm{M}}_{22} - \hat{\mathrm{M}}_{21}\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12}$ is approximately zero so that we can speak of the approximation

$$\Pi_1 \hat{\mathrm{M}}(d) \Pi_2 \approx \begin{bmatrix} \hat{\mathrm{M}}_{11} \\ \hat{\mathrm{M}}_{21} \end{bmatrix} \begin{bmatrix} \mathrm{I}_{r(d)} & \hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12} \end{bmatrix}.$$

Subsequently,

$$\hat{\mathrm{N}}(d) := \Pi_2 \begin{bmatrix} -\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12} \\ \mathrm{I}_{d_\Sigma^2} \end{bmatrix} \tag{3.14}$$

is a numerical approximation to the right null space of $\hat{\mathrm{M}}(d)$, and it is additionally desirable in this setting that the entries of $\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12}$ remain small in absolute value to ensure stability of the representation, in which case, one has a strong rank-revealing LU-factorization [37].

3.2.2. Cauchy representation of the null space.

Let

$$\tilde{\mathrm{N}} := -\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12} \in \mathbb{C}^{r(d)\times d_\Sigma^2}. \tag{3.15}$$

If (2.8) is exactly satisfied, the columns of

$$\mathrm{N}(d) = \Psi \hat{\mathrm{N}}(d) = \Psi \Pi_2 \begin{bmatrix} \tilde{\mathrm{N}} \\ \mathrm{I}_{d_\Sigma^2} \end{bmatrix} \tag{3.16}$$

provide a numerical basis for the right null space of the original Macaulay matrix (2.4). Direct application of Gaussian elimination on $\hat{\mathrm{M}}(d)$ will not result in any fast algorithm to generate (3.15). To achieve that, one has to take advantage of the fact that (3.15) is also Cauchy-like, with a displacement rank equal to that of the original Macaulay matrix. To verify this property, partition $\Pi_2 = \begin{bmatrix} \Pi_{2,a} & \Pi_{2,b} \end{bmatrix}$ with $\Pi_{2,a} \in \mathbb{R}^{n(d)\times r(d)}$ and $\Pi_{2,b} \in \mathbb{R}^{n(d)\times d_\Sigma^2}$. It can shown that the augmented matrix

$$\begin{bmatrix} \Pi_1 & \\ & \mathrm{I}_{r(d)} \end{bmatrix} \begin{bmatrix} \hat{\mathrm{M}}(d) \\ \Pi_{2,a}^\top \end{bmatrix} \begin{bmatrix} \Pi_{2,a} & \Pi_{2,b} \end{bmatrix} = \left[\begin{array}{cc} \hat{\mathrm{M}}_{11} & \hat{\mathrm{M}}_{12} \\ \hat{\mathrm{M}}_{21} & \hat{\mathrm{M}}_{22} \\ \hline \mathrm{I}_{r(d)} & \mathbb{0}_{r\times d_\Sigma^2} \end{array} \right], \tag{3.17}$$

satisfies the displacement equation

$$\left[\begin{array}{c|c} \operatorname{diag}(\boldsymbol{\kappa}) & \\ \hline & \operatorname{diag}(\boldsymbol{\xi}) \end{array}\right] \left[\begin{array}{c|c} \hat{\mathrm{M}}_{11} & \hat{\mathrm{M}}_{12} \\ \hat{\mathrm{M}}_{21} & \hat{\mathrm{M}}_{22} \\ \hline \mathrm{I}_{r(d)} & \mathbb{0}_{r(d)\times d_\Sigma^2} \end{array}\right] -$$

$$\left[\begin{array}{c|c} \hat{\mathrm{M}}_{11} & \hat{\mathrm{M}}_{12} \\ \hat{\mathrm{M}}_{21} & \hat{\mathrm{M}}_{22} \\ \hline \mathrm{I}_{r(d)} & \mathbb{0}_{r(d)\times d_\Sigma^2} \end{array}\right] \left[\begin{array}{c|c} \operatorname{diag}(\boldsymbol{\xi}) & \\ \hline & \operatorname{diag}(\boldsymbol{\eta}) \end{array}\right] = \left[\begin{array}{c} \mathrm{U}_a \\ \mathrm{U}_b \\ \hline \mathbb{0}_{r(d)\times S(\Delta d+1)} \end{array}\right] \begin{bmatrix} \mathrm{V}_a^* & \mathrm{V}_b^* \end{bmatrix},$$

with $\boldsymbol{\kappa} \in \mathbb{C}^{m(d)}$, $\boldsymbol{\xi} \in \mathbb{C}^{r(d)}$, $\boldsymbol{\eta} \in \mathbb{C}^{d_\Sigma^2}$, $\mathrm{V}_a \in \mathbb{C}^{r(d)\times\rho(d)}$, $\mathrm{V}_b \in \mathbb{C}^{d_\Sigma^2\times\rho(d)}$, $\mathrm{U}_a \in \mathbb{C}^{r(d)\times\rho(d)}$, and $\mathrm{U}_b \in \mathbb{C}^{(m(d)-r(d))\times\rho(d)}$ given by

$$\boldsymbol{\kappa} = \Pi_1\boldsymbol{\mu}, \quad \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{bmatrix} = \Pi_2\boldsymbol{\nu}, \quad \begin{bmatrix} \mathrm{U}_a \\ \mathrm{U}_b \end{bmatrix} = \Pi_1\mathrm{U}, \quad \begin{bmatrix} V_a \\ V_b \end{bmatrix} = \Pi_2^\top V.$$

Since, by row-reduction, we have the equivalence

$$\begin{bmatrix} \hat{\mathrm{M}}_{11} & \hat{\mathrm{M}}_{12} \\ \hat{\mathrm{M}}_{21} & \hat{\mathrm{M}}_{22} \\ \mathrm{I}_{r(d)} & \mathbb{0}_{r(d)\times d_\Sigma^2} \end{bmatrix} \sim \begin{bmatrix} \hat{\mathrm{M}}_{11} & \hat{\mathrm{M}}_{12} \\ \mathbb{0}_{(m(d)-r(d))\times r(d)} & \hat{\mathrm{M}}_{22} - \hat{\mathrm{M}}_{21}\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12} \\ \mathbb{0}_{r(d)\times r(d)} & \tilde{\mathrm{N}} \end{bmatrix},$$

further algebraic deductions would reveal that (3.15) satisfies the displacement equation

$$\operatorname{diag}(\boldsymbol{\xi})\tilde{\mathrm{N}} - \tilde{\mathrm{N}}\operatorname{diag}(\boldsymbol{\eta}) = \left(-\hat{\mathrm{M}}_{11}^{-1}\mathrm{U}_a\right)\left(\mathrm{V}_b - \tilde{\mathrm{N}}^*\mathrm{V}_a\right) =: \mathrm{RS}^*. \tag{3.18}$$

If one chooses $\{\varphi_j\}_{j=1}^{d+1}$ such that $\boldsymbol{\nu}$ only has distinct entries, $\boldsymbol{\xi} \in \mathbb{C}^{r(d)}$ will have no entries in common with $\boldsymbol{\eta} \in \mathbb{C}^{d_\Sigma^2}$. The displacement operator in (3.18) is subsequently invertible (see Footnote 7), and hence,

$$\tilde{\mathrm{N}} = \mathscr{C}\left(\boldsymbol{\xi}, \boldsymbol{\eta}, \mathrm{R}, \mathrm{S}\right), \tag{3.19}$$

with $\mathrm{R} \in \mathbb{C}^{r(d)\times\rho(d)}$ and $\mathrm{S} \in \mathbb{C}^{d_\Sigma^2\times\rho(d)}$, comprises a valid compact representation for (3.15).

3.2.3. Schur algorithm for Cauchy-like matrices. The LU-factorization of a Cauchy-like matrix can be determined efficiently using the Schur algorithm [29]. The Schur algorithm relies on the *key* property that the Schur complement of a Cauchy-like matrix is also Cauchy-like, with the displacement being equal to that of the original matrix; see e.g., [24, Theorem 12.1.1] for a precise statement. Subsequently, each step of Gaussian elimination can be performed efficiently by updating the entries of the generators (instead of the dense matrix itself). With the foregoing discussions in Subsection 3.2.2, the Schur algorithm may also be adapted to determine the generators of (3.19), and hence, obtain a compact representation for (3.15). The details are given below.

ALGORITHM 1 (Modified Schur algorithm for null space of Cauchy-like matrix).
IN: $\hat{\mathrm{M}} = \mathscr{C}(\mathrm{U}, \mathrm{V}, \boldsymbol{\mu}, \boldsymbol{\nu})$, $\epsilon > 0$
OUT: $\tilde{\mathrm{N}} = \mathscr{C}(\mathrm{R}, \mathrm{S}, \boldsymbol{\xi}, \boldsymbol{\eta}), \Pi_2$

1. *Initialize*

$$\Pi_1^{(0)} = \mathrm{I}_{m(d)}, \qquad \boldsymbol{\mu}^{(0)} = \boldsymbol{\mu}, \qquad \mathrm{U}^{(0)} = \mathrm{U},$$
$$\Pi_2^{(0)} = \mathrm{I}_{n(d)}, \qquad \boldsymbol{\nu}^{(k)} = \boldsymbol{\nu}, \qquad \mathrm{V}^{(0)} = \mathrm{V},$$

and set $\hat{\mathrm{M}}^{(0)} := \mathscr{C}\left(\boldsymbol{\mu}^{(0)}, \boldsymbol{\nu}^{(0)}, \mathrm{U}^{(0)}, \mathrm{V}^{(0)}\right) = \hat{\mathrm{M}}(d)$.

2. *For* $k = 1, 2, \ldots, \min\{m(d), n(d)\}$, *repeat the following steps (see Subsection 3.2.4 for more details):*
   (a) *Given a certain (rank-revealing) pivoting strategy, pivot the* (i_k, j_k)*-th entry of* $\hat{\mathrm{M}}^{(k-1)}$ *with* $i_k, j_k \geqslant k$ *to the* (k,k)*-th position. That is, if* $\Gamma_i \in \mathbb{R}^{m(d)\times m(d)}$ *and* $\Xi_i \in \mathbb{R}^{n(d)\times n(d)}$ *denote the corresponding row and column interchange permutations to achieve this pivoting action, then*

$$\Pi_1^{(k)} = \Gamma_k \Pi_1^{(k-1)}, \qquad \boldsymbol{\mu}^{(k)} = \Gamma_k \boldsymbol{\mu}^{(k-1)}, \qquad \tilde{\mathrm{U}}^{(k)} = \Gamma_k \mathrm{U}^{(k-1)},$$
$$\Pi_2^{(k)} = \Pi_2^{(k-1)} \Xi_k, \qquad \boldsymbol{\nu}^{(k)} = \Xi_k \boldsymbol{\nu}^{(k-1)}, \qquad \tilde{\mathrm{V}}^{(k)} = \Xi_k \mathrm{V}^{(k-1)},$$

   and $\tilde{\mathrm{M}}^{(k)} := \mathscr{C}\left(\boldsymbol{\mu}^{(k)}, \boldsymbol{\nu}^{(k)}, \tilde{\mathrm{U}}^{(k)}, \tilde{\mathrm{V}}^{(k)}\right) = \Pi_1^{(k)} \hat{\mathrm{M}}^{(k-1)} \Pi_2^{(k)}$.

   (b) *Evaluate* $\alpha_k = \tilde{\boldsymbol{u}}_k^{(k)*} \tilde{\boldsymbol{v}}_k^{(k)} / (\mu_k^{(k)} - \nu_k^{(k)})$,

$$\boldsymbol{w}_k = \begin{bmatrix} \frac{\tilde{\boldsymbol{u}}_1^{(k)*} \tilde{\boldsymbol{v}}_k^{(k)}}{\nu_1^{(k-1)} - \nu_k^{(k)}}, \\ \vdots \\ \frac{\tilde{\boldsymbol{u}}_{k-1}^{(k)*} \tilde{\boldsymbol{v}}_k^{(k)}}{\nu_{k-1}^{(k-1)} - \nu_k^{(k)}} \end{bmatrix}, \quad \boldsymbol{g}_k = \begin{bmatrix} \frac{\tilde{\boldsymbol{u}}_{k+1}^{(k)*} \tilde{\boldsymbol{v}}_k^{(k)}}{\mu_{k+1}^{(k)} - \nu_k^{(k)}}, \\ \vdots \\ \frac{\tilde{\boldsymbol{u}}_{m(d)}^{(k)*} \tilde{\boldsymbol{v}}_k^{(k)}}{\mu_{m(d)}^{(k)} - \nu_k^{(k)}} \end{bmatrix}, \quad \boldsymbol{h}_k = \begin{bmatrix} \frac{\tilde{\boldsymbol{v}}_{k+1}^{(k)*} \tilde{\boldsymbol{u}}_k^{(k)}}{\mu_k^{(k)} - \nu_{k+1}^{(k)}} \\ \vdots \\ \frac{\tilde{\boldsymbol{v}}_{n(d)}^{(k)*} \tilde{\boldsymbol{u}}_k^{(k)}}{\mu_k^{(k)} - \nu_{n(d)}^{(k)}}, \end{bmatrix},$$

   to form the Gauss transforms

$$\mathrm{G}_k = \mathrm{I}_{m(d)} - \frac{1}{\alpha_k} \begin{bmatrix} \boldsymbol{w}_k \\ \alpha_k + 1 \\ \boldsymbol{g}_k \end{bmatrix} \boldsymbol{e}_{k,m(d)}^\top, \quad \mathrm{H}_k = \mathrm{I}_{n(d)} - \frac{1}{\bar{\alpha}_k} \begin{bmatrix} \mathbb{0}_k \\ \boldsymbol{h}_k \end{bmatrix} \boldsymbol{e}_{k,n(d)}^\top,$$

   and perform Gaussian elimination on the generators

$$\mathrm{U}^{(k)} = \mathrm{G}_k \tilde{\mathrm{U}}^{(k)}, \quad \mathrm{V}^{(k)} = \mathrm{H}_k \tilde{\mathrm{V}}^{(k)},$$

   to subsequently define $\hat{\mathrm{M}}^{(k)} := \mathscr{C}\left(\boldsymbol{\mu}^{(k)}, \boldsymbol{\nu}^{(k)}, \mathrm{U}^{(k)}, \mathrm{V}^{(k)}\right)$.

   (c) *Let*

$$\boldsymbol{\xi}^{(k)} = \boldsymbol{\nu}_{1:k}^{(k)}, \qquad \mathrm{R}^{(k)} = \mathrm{U}_{1:k,:}^{(k)},$$
$$\boldsymbol{\eta}^{(k)} = \boldsymbol{\nu}_{k+1:n(d)}^{(k)}, \qquad \mathrm{S}^{(k)} = \mathrm{V}_{:,k+1:n(d)}^{(k)},$$

   and set $\tilde{\mathrm{N}}^{(k)} := \mathscr{C}\left(\boldsymbol{\xi}^{(k)}, \boldsymbol{\eta}^{(k)}, \mathrm{R}^{(k)}, \mathrm{S}^{(k)}\right)$.

   (d) *Check whether*

$$\left\| \hat{M}_{k+1:m(d),k+1:n(d)}^{(k)} \right\|_{\mathrm{F}} \leqslant \epsilon. \tag{3.20}$$

   If (3.20) *is indeed satisfied, break the loop and proceed to step 3.*

3. *Set* $\tilde{\mathrm{N}} = \tilde{\mathrm{N}}^{(k)}$, *and hence,* $\boldsymbol{\xi} = \boldsymbol{\xi}^{(k)}$, $\boldsymbol{\eta} = \boldsymbol{\eta}^{(k)}$, $\mathrm{R} = \mathrm{R}^{(k)}$, $\mathrm{S} = \mathrm{S}^{(k)}$, $\Pi_1 = \Pi_1^{(k)}$, *and* $\Pi_2 = \Pi_2^{(k)}$.

3.2.4. Efficient complete pivoting and evaluation of stopping criteria. The procedure outlined in Subsection 3.2.3 requires further elaboration on two aspects: (i) how to exactly pivot the entries of (3.6) such that a rank-revealing LU-factorization is obtained, and (ii) how to efficiently evaluate the stopping criterion (3.20) without explicitly forming the Schur complement and computing its norm.

It is well-known that, in exact arithmetic, Gaussian elimination with complete pivoting always reveals the rank of a matrix. Although one cannot ensure that this property persists under floating point arithmetic (see examples in [37,45]), it is plausible to assume that complete pivoting should work decently in practice, at least for the matrices considered in this paper. However, direct application of complete pivoting by searching through all the matrix entries is prohibitively expensive and destroys the asymptotic complexity gains that one would achieve with the Schur algorithm.

Nonetheless, it turns out that a suitable pivot can directly be found from the generators of the Cauchy-like matrix if one relaxes the requirement to always find the largest magnitude matrix entry. This method, originally introduced by Ming Gu, is based upon a fundamental observation made in [26, Lemma 3.1] which, restated for matrix $\hat{\mathrm{M}}^{(k-1)} \in \mathbb{C}^{m(d)\times n(d)}$ in Algorithm 1, says that if j_k^* denotes the column position of the column with maximum 2-norm in $\mathrm{U}^{(k-1)}_{k:m(d),:}\mathrm{V}^{(k-1)^*}$, then the following lower bound is satisfied:

$$\max_{k\leqslant i\leqslant m(d)} \left|\hat{m}^{(k)}_{ij_k^*}\right| \geqslant \frac{1}{K\sqrt{n(d)-k}} \max_{\substack{k\leqslant i\leqslant m(d)\\ k\leqslant j\leqslant n(d)}} \left|\hat{m}^{(k)}_{ij}\right|,\ K := \max_{\substack{k\leqslant i,\imath\leqslant m(d)\\ k\leqslant j,\jmath\leqslant n(d)}} \frac{\left|\mu_i^{(k)} - \nu_j^{(k)}\right|}{\left|\mu_\imath^{(k)} - \nu_\jmath^{(k)}\right|}. \tag{3.21}$$

That is, the j_k^*'th column of $\hat{\mathrm{M}}^{(k)}$ already contains a sufficiently large pivot. Furthermore, this column can be found rather efficiently (i.e., without breaking the complexity gains made by the Schur algorithm) provided the columns of $\mathrm{U}^{(k-1)}_{k:m(d),:}$ are orthonormal[9]. A similar statement can also be made for the stopping criterion (3.20), since [26, Lemma 3.1] also establishes the bound

$$\left\|\hat{M}^{(k)}_{k+1:m(d),k+1:n(d)}\right\|_{\mathrm{F}} \leqslant K\sqrt{(n(d)-k-1)(m(d)-k-1)} \max_{k+1\leqslant i\leqslant m(d)} \left|\hat{m}^{(k)}_{ij^*_{k+1}}\right|.$$

In the subsequent section, it is explained how $\mathrm{U}^{(k-1)}_{k:m(d),:}$ can be kept orthonormal throughout the execution of the Schur algorithm.

3.2.5. Re-orthonormalization procedure. The orthormality of $\mathrm{U}^{(k)}_{k+1:m(d),:}$ is destroyed in step 2(b) of Algorithm 1 when the Gauss-updates are performed. To find a suitable pivot, a re-orthonormalization procedure must be incorporated in this step to maintain orthonormality of $\mathrm{U}^{(k)}_{k+1:m(d),:}$. A naive approach, which would break the asymptotic complexity of the algorithm, is to compute a QR-decomposition $\mathrm{U}^{(k)}_{k+1:m(d),:} = \mathrm{Q}^{(k)}\mathrm{B}^{(k)}$ from scratch at each iteration so that

$$\mathrm{U}^{(k)} \leftarrow \begin{bmatrix} \mathrm{U}^{(k)}_{1:k,:}\left(\mathrm{B}^{(k)}\right)^{-1} \\ \mathrm{Q}^{(k)} \end{bmatrix}, \quad \mathrm{V}^{(k)}_{k+1:n(d),:} \leftarrow \mathrm{V}^{(k)}_{k+1:n(d),:}\left(\mathrm{B}^{(k)}\right)^*. \tag{3.22}$$

Instead, the re-orthonormalization must be achieved through clever updating strategies. Assuming orthonormality[10] of $\tilde{\mathrm{U}}^{(k)}_{k:m(d),:}$, step 2(b) of Algorithm 1 can be replaced

[9] In which case, it suffices to just compute the 2-norms of the rows of $\mathrm{V}^{(k-1)}$.
[10] This property is already satisfied at initiation of Algorithm 1!

by Algorithm 2.

Unfortunately, Algorithm 2 by itself will introduce numerical issues. Even though $\mathrm{U}^{(k)}_{k+1:m(d),:}$ and $\mathrm{V}^{(k)}_{k+1:n(d),:}$ are computed stably, $\mathrm{U}^{(k)}_{1:k,:}$ loses accuracy[11] as the iterations proceed if $\mathrm{B}^{(k)}$ in (3.22) is close to singular. This is a cause of concern since $\mathrm{U}^{(k)}_{1:k,:}$ is a key term in the construction of $\tilde{\mathrm{N}}$. One may overcome this challenge by running two versions of Algorithm 1 in parallel. Since only $\mathrm{U}^{(k)}_{k+1:m(d),:}$ and $\mathrm{V}^{(k)}_{k+1:n(d),:}$ are needed in the pivot selection, the first version will use Algorithm 2 to *solely* find a pivot. For the second version, Algorithm 1 is run without Algorithm 2 to avoid loss of accuracy in $\mathrm{U}^{(k)}_{1:k,:}$. This will increase the cost of running the entire algorithm by a factor two, but will not break its asymptotic complexity. A more efficient remedy to this problem is an open question.

Algorithm 2 (Gauss-update step with orthonormalization).

In: $\tilde{\mathrm{U}}^{(k)}$ *with* $\tilde{\mathrm{U}}^{(k)*}_{k:m(d),:}\tilde{\mathrm{U}}^{(k)}_{k:m(d),:} = \mathrm{I}_{\rho(d)}$, $\tilde{\mathrm{V}}^{(k)}$

Out: $\mathrm{U}^{(k)}$ *with* $\mathrm{U}^{(k)*}_{k+1:m(d),:}\mathrm{U}^{(k)}_{k+1:m(d),:} = \mathrm{I}_{\rho(d)}$, $\mathrm{V}^{(k)}$

1. *Make* $\tilde{\mathrm{U}}^{(k)}_{k,:}$ *equal to* $c\boldsymbol{e}^{\top}_{1,\rho(d)}$ *for some* $c \in \mathbb{C}$ *by using a suitable Householder transformation* F*, i.e.,*

$$\tilde{\mathrm{U}}^{(k)} \leftarrow \tilde{\mathrm{U}}^{(k)}\mathrm{F}^*, \quad \tilde{\mathrm{V}}^{(k)} \leftarrow \tilde{\mathrm{V}}^{(k)}\mathrm{F}^*.$$

.

2. *Perform the Gauss-update step with* G_k *and* H_k *computed as in step 2 of Algorithm 1,*

$$\mathrm{U}^{(k)} = \mathrm{G}_k\tilde{\mathrm{U}}^{(k)}, \quad \mathrm{V}^{(k)} = \mathrm{H}_k\tilde{\mathrm{V}}^{(k)},$$

which now only modifies the first column in $\mathrm{U}^{(k)}$ *due to the re-assignment in step 1.*

3. *Reorthogonalize the first column of* $\mathrm{U}^{(k)}_{(k+1):m(d),:}$ *by performing the updates*

$$\begin{aligned}\boldsymbol{b}^{(k)} &= \left(\mathrm{U}^{(k)}_{k+1:m(d),2:r(d)}\right)^* \mathrm{U}^{(k)}_{k+1:m(d),1},\\ \mathrm{U}^{(k)}_{:,1} &\leftarrow \mathrm{U}^{(k)}_{:,1} - \mathrm{U}^{(k)}_{:,2:r(d)}\boldsymbol{b}^{(k)},\\ \mathrm{V}^{(k)}_{:,2:r(d)} &\leftarrow \mathrm{V}^{(k)}_{:,2:r(d)} + \mathrm{V}^{(k)}_{:,1}\left(\boldsymbol{b}^{(k)}\right)^*,\end{aligned}$$

and note that $\mathrm{U}^{(k)}_{k+1:m(d),2:r(d)}$ *is already orthonormal due to step 2.*

4. *Normalize the first column of* $\mathrm{U}^{(k)}_{k+1:m(d),:}$ *by performing the updates*

$$\mathrm{U}^{(k)}_{:,1} \leftarrow \frac{\mathrm{U}^{(k)}_{:,1}}{\left\|\mathrm{U}^{(k)}_{k+1:m(d),1}\right\|_2}, \quad \mathrm{V}^{(k)}_{:,1} \leftarrow \mathrm{V}^{(k)}_{:,1}\left\|\mathrm{U}^{(k)}_{k+1:m(d),1}\right\|_2.$$

Remark 3.2. In step 4, the norm of $\mathrm{U}^{(k)}_{k+1:m(d),1}$ may become zero in the course of the execution of the algorithm. This means that $\hat{\mathrm{M}}^{(k)}_{k+1:m(d),k+1:n(d)}$ is a matrix of displacement rank smaller than $\hat{\mathrm{M}}^{(k)}_{k:m(d),k:n(d)}$. Instead of normalizing $\mathrm{U}^{(k)}_{k+1:m(d),1}$, we

[11] A property that has also been observed in practice in our initial experiments.

can drop this first column along with the first column of $\mathrm{V}^{(k)}_{k+1:n(d),:}$ and continue with the rest of the columns. Numerically, these columns can be dropped if the norm is close to machine precision.

Remark 3.3. Step 3 should be done in a numerically stable manner by applying Gram–Schmidt twice [22].

3.3. Summary of algorithm and complexity analysis. Returning back to Figure 2, the following algorithm is proposed to determine a numerical null space $\mathrm{N}(d)$ of the Macaulay matrix (2.4) associated with the polynomial system (1.1).

ALGORITHM 3 (Fast null space of Macaulay matrix).
IN: $\mathrm{M}(d)$
OUT: $\mathrm{N}(d)$

1. *Construct the compact representation of* $\hat{\mathrm{M}}(d)$*, as specified in* (3.8) *in terms of the generators* $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, U, V *defined in* (3.10) *and* (3.12)*, respectively. Use FFTs to accelerate the construction of* V*. Furthermore, ensure that* $\{\varphi_i\}_{j=1}^{d+1}$ *are chosen such that: (i) the entries of* $\boldsymbol{\nu}$ *are all distinct, and (ii) do not coincide with any entry in* $\boldsymbol{\eta}$*. Practical choices for* $\{\varphi_i\}_{j=1}^{d+1}$ *are discussed in Subsection 4.1.1.*
2. *Given the generators of* $\hat{\mathrm{M}}(d)$ *and a user-specified tolerance* $\epsilon > 0$*, run Algorithm 1 while maintaining two copies of* U *and* V*. Perform the Schur updates on the first copy through Algorithm 2 and obtain the pivot from* V*. For the second copy, perform the update as in Algorithm 1 and use this copy to obtain* $\hat{\mathrm{N}}$ *as specified in* (3.19) *in terms of the generators* $\boldsymbol{\xi}$, $\boldsymbol{\eta}$, R, S.
3. *Evaluate the expression* (3.16) *by using FFTs and taking advantage of the block-diagonal structure in* Φ*, as defined in* (3.9)*.*

Estimates on the number of floating point operations involved for the first and last step are $\mathcal{O}(S \cdot d_\Sigma \cdot \Delta d \cdot d \log d)$ and $\mathcal{O}(d_\Sigma^2 \cdot d^2 \log d)$, respectively. The second step is by far the most expensive and dominates the null space computation. A careful analysis reveal that the Gaussian elimination in step 2(b) and the orthogonalization procedure are the main computational bottlenecks in Algorithm 1. The per iteration cost involves at most $\mathcal{O}(S^2 d^3)$ floating point operations, and if condition (2.8) is satisfied, it is expected that $r(d)$ steps will be required, leading to a total complexity of $\mathcal{O}(r(d) \cdot S^2 d^3)$. Together with the bound on the degree of regularity (2.11), one further deduces that the complexity of Algorithm 1 is $\mathcal{O}(S^2 d_\Sigma^5)$ for a Macaulay matrix of degree $d \leqslant 2d_\Sigma - 2$. Since[12] typically $S \ll d_\Sigma$, one attains overall an $\mathcal{O}(d_\Sigma^5)$ algorithm for determining a null space from where one can further deduce the roots of the system (e.g., using the method described in Subsection 2.3). We may compare this complexity with that of obtaining a null space basis from a singular value decomposition. To produce the singular values and right singular vectors of $\mathrm{M}(d)$ using the Golub-Reinsch algorithm will involve $\mathcal{O}(4Sd^6 + 8d^6)$ floating point operations [24, Figure 8.6.1]. Hence, a complexity reduction from $\mathcal{O}(d_\Sigma^6)$ to $\mathcal{O}(d_\Sigma^5)$ is achieved.

4. Numerical experiments.

In the subsequent sections, we empirically evaluate the accuracy (Subsection 4.2) and computational complexity (Subsection 3.3) of the developed algorithm[13].

[12] Furthermore, note that for highly overdetermined systems, it is possible to apply sampling on the rows to exploit redundancy; see e.g., [41].

[13] Algorithm 3 was implemented in the Julia programming language and can be obtained by contacting the authors of this paper. All experiments were run on a laptop with 32 GB RAM and

4.1. Experiment setup. To test our algorithm, we generate two polynomials of degree d_Σ with standard normal random coefficients. The parameter d is *always* set to $2d_\Sigma - 2$; the upper bound on the degree of regularity d^*. To evaluate the error, we use the metric

$$\epsilon := \frac{\|\mathrm{M}(d)\mathrm{Q}\|_2}{\|\mathrm{M}(d)\|_2}, \tag{4.1}$$

where $\mathrm{Q} \in \mathbb{C}^{n(d)\times d_\Sigma^2}$ refers to an orthonormal basis for col $\mathrm{N}(d)$ obtained from a QR-decomposition. For a fair comparison, especially in the presence of noise, this error should be compared with its lower bound, namely $\epsilon_{\min} = \frac{\sigma_{r(d)+1}}{\sigma_1}$, which thus only depends on the singular values of the Macaulay matrix.

To study the behavior of our algorithm, we compare our method with easier methods by removing layers of complexity one-by-one. All these methods are expected to have equal or slightly better stability, but are asymptotically slower to compute (i.e., $\mathcal{O}(d_\Sigma^6)$ instead of $\mathcal{O}(d_\Sigma^5)$).

- SVD on $\mathrm{M}(d)/\hat{\mathrm{M}}(d)$: computing the SVD on the dense Macaulay matrix $\mathrm{M}(d)$ or the dense Cauchy-like matrix $\hat{\mathrm{M}}(d)$. Note that this method's error is always (approximately) equal to the lower bound $\epsilon_{\min}$.
- GECP on $\mathrm{M}(d)$: Gaussian elimination with complete pivoting on the dense Macaulay matrix $\mathrm{M}(d)$.
- GECP on $\hat{\mathrm{M}}(d)$: Gaussian elimination with complete pivoting on the dense Cauchy-like matrix $\hat{\mathrm{M}}(d)$.
- GECP on $\mathscr{C}$: the Schur algorithm with complete pivoting, or in other words, Gaussian elimination with complete pivoting on the compact representation of the Cauchy-like matrix $\hat{\mathrm{M}}(d)$. This compact representation is denoted as $\mathscr{C}$ in this section and was explained in Subsection 3.1.2.
- GEAP on $\mathscr{C}$: the Schur algorithm with approximate complete pivoting as explained in Subsection 3.2.4. This is the method presented in this paper (Algorithm 3) and the only method with complexity $\mathcal{O}(d_\Sigma^5)$ instead of $\mathcal{O}(d_\Sigma^6)$ (as discussed in Subsection 4.3).

4.1.1. Choice of φ. The generators $\{\varphi_j\}_{j=1}^{d+1}$, introduced in Subsection 3.1.1, should be chosen in such a way that singularity of operator (3.2) is avoided. The operator is singular for the Macaulay matrix if, for any i, j, $\mu_i = \nu_j$ and for the null space if $\xi_i = \eta_j$. From a numerical point of view, if the operator is close to singular, the problem will become ill-conditioned, leading to a loss of stability. Because of this, maximizing the differences $|\mu_i - \nu_j|$ and $|\xi_i - \eta_j|$ for all i, j seems to be a sensible criterion, corroborated by the experiment in Subsection 4.2.1. As the partitioning of $\boldsymbol{\nu}$ into $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ is not known a priori, we instead maximize the difference $|\nu_i - \nu_j|$ for all i, j where $i \neq j$.

In these experiments, a greedy method was employed to choose $\{\varphi_j\}_{j=1}^{d+1}$ to obtain a well-conditioned Cauchy representation. At iteration k, the optimal φ_k is chosen to maximize

$$\min\{\min_{i,j} |\mu_i - \nu_j^{(k)}|, \min_{i,j,i\neq j} |\nu_i^{(k)} - \nu_j^{(k)}|\},$$

where $\boldsymbol{\nu}^{(k)}$ only contains $\{\varphi_i^{1/i}\Omega_i\}_{i=\Delta d+1}^{\Delta d+2-k}$. This greedy algorithm requires $\mathcal{O}(d^3)$ flops.

an AMD Ryzen 7 PRO 5850U CPU @ 1.90 GHz.

4.2. Empirical analysis of stability. We first study the stability of the algorithm in the square case (Subsection 4.2.1) where condition Equation (2.8) is satisfied by default. Then, we study the effect of noise on overdetermined systems (Subsection 4.2.2) where condition Equation (2.8) is satisfied only approximately.

4.2.1. Square systems. Table 1 shows how the error grows for increasing degree d_Σ and for the different methods.

	d_Σ				
	2	4	8	16	32
SVD on $\mathrm{M}(d)$	2.23e-16	3.75e-16	5.70e-16	7.94e-16	9.51e-16
SVD on $\hat{\mathrm{M}}(d)$	2.57e-16	4.77e-16	7.54e-16	9.97e-16	1.15e-15
GECP on $\mathrm{M}(d)$	1.40e-16	3.11e-16	8.33e-16	1.02e-14	1.40e-13
GECP on $\hat{\mathrm{M}}(d)$	2.08e-16	4.65e-16	1.03e-15	9.73e-15	1.21e-13
GECP on $\mathscr{C}$	4.35e-16	1.51e-15	1.35e-14	1.72e-13	2.81e-12
GEAP on $\mathscr{C}$	4.21e-16	3.63e-15	3.88e-14	3.19e-13	4.48e-12

Table 1: Median error for different methods (see Subsection 4.1 for an explanation of the abbreviations) and degrees d_Σ over 100 runs. We see that the error arises mostly from using an LU-factorization instead of an SVD and working on the compact representation $\mathscr{C}$ instead of $\mathrm{M}(d)$ or $\hat{\mathrm{M}}(d)$.

The two biggest sources of error are switching from an SVD to a LU-factorization, as expected, and working on the compact representation $\mathscr{C}$ instead of the full $\hat{\mathrm{M}}(d)$. In Table 2, results with purposefully poorly-chosen generators of the Cauchy representation are shown. These corroborate the reasoning in Subsection 4.1.1, namely that the minimum gap of the generators $\gamma_{\min}$ affects the numerical stability, due to a division by a small difference of the generators $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. The results in Table 2 seem to suggest an inverse proportional relation between the error and the minimum gap $\gamma_{\min}$, namely

$$\epsilon \sim \frac{\epsilon_{\mathrm{mach}}}{\gamma_{\min}} \quad \text{where} \quad \gamma_{\min} := \frac{\min\left\{\min_{i,j}|\mu_i - \nu_j|, \min_{i,j,i\neq j}|\nu_i - \nu_j|\right\}}{\max\left\{\max_{i,j}|\mu_i - \nu_j|, \max_{i,j,i\neq j}|\nu_i - \nu_j|\right\}}. \tag{4.2}$$

4.2.2. Noisy overdetermined case. In this experiment we first generate two polynomials as above and then a third polynomial as a random linear combination of the two first generated polynomials. The degree d_Σ is fixed to 16. Then additive Gaussian noise is added on the coefficients of the polynomials to obtain a fixed signal-to-noise ratio, measured as $\|\mathrm{M}(d)\|_{\mathrm{F}}^2 / \|\mathrm{M}_{\mathrm{noisy}}(d) - \mathrm{M}(d)\|_{\mathrm{F}}^2$. Figure 4 shows the results. The LU-based methods initially stay close to the SVD (and thus $\epsilon_{\min}$), but as the noise rises, worsen in performance. With approximate complete pivoting this happens slightly earlier than with (exact) complete pivoting.

To decrease this error in the end, one could potentially look at iterative refinement techniques, which could push the accuracy of LU-based methods further towards that of SVD without paying a price for overall complexity. This was not further investigated here.

4.3. Algorithm complexity. As stated in Subsection 3.3, the presented approach reduces the computational complexity from $\mathcal{O}(d_\Sigma^6)$ to $\mathcal{O}(d_\Sigma^5)$. This was checked

		Method		
φ-generation	$\gamma_{\min}$	GECP on $\hat{\mathrm{M}}(d)$	GECP on $\mathscr{C}$	GEAP on $\mathscr{C}$
Greedy	1.01e-03	9.73e-15	1.72e-13	3.19e-13
Random	6.80e-06	9.37e-15	6.01e-12	1.20e-11
Fixed gap	1.00e-04	9.21e-15	1.62e-12	3.22e-12
Fixed gap	1.00e-06	9.00e-15	1.56e-10	3.25e-10
Fixed gap	1.00e-08	9.28e-15	1.57e-08	3.34e-08

Table 2: Median error ϵ with different strategies for generating the generator ν over 100 runs. The "Greedy" strategy was presented in Subsection 4.1.1, "Random" generates uniform random $\{\varphi_j\}_{j=1}^{d+1}$ on the unit circle, while "Fixed gap" selects $\{\varphi_j\}_{j=1}^{d+1}$ such that the smallest $d+1$ gaps are all equal to a fixed quantity. We see that the minimum gap ($\gamma_{\min}$ as defined in Equation (4.2)) has no impact on the error of Gaussian elimination with complete pivoted on the full Cauchy matrix $\hat{\mathrm{M}}(d)$, while it is inversely correlated with the error of Gaussian elimination on the compact representation of the Cauchy matrix $\mathscr{C}$ for both complete and approximate complete pivoting.

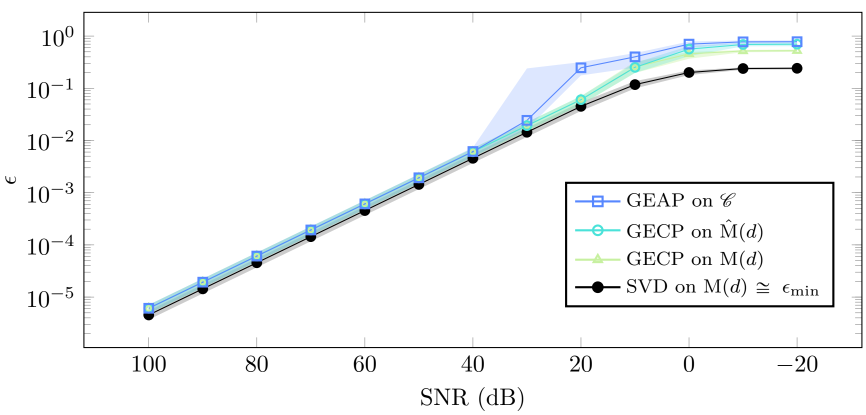
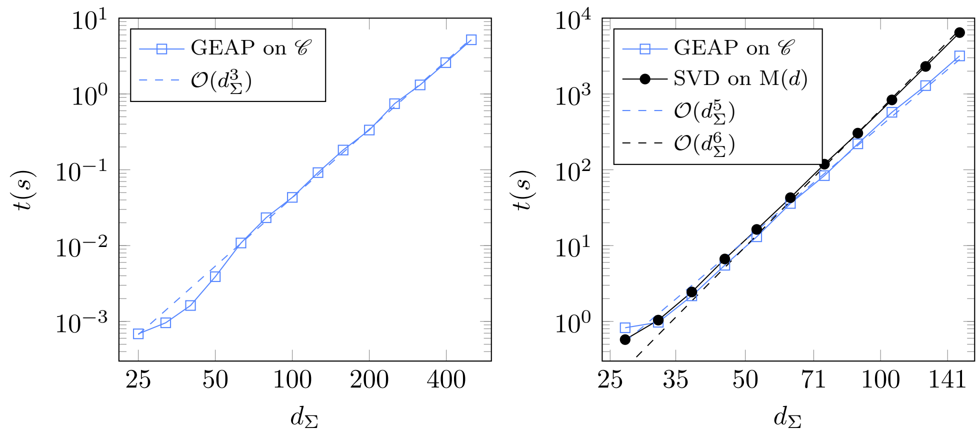


Fig. 4: Median error ϵ (with 25% and 75% quantiles around) for different signal-to-noise levels and methods over 1000 experiments (see Subsection 4.1 for an explanation of the abbreviations). GECP on $\mathscr{C}$ was not drawn as this was identical to GECP on $\hat{\mathrm{M}}(d)$. We see that GECP on whichever representation (compact Cauchy or full) has similar accuracy, only marginally worse than the best method (SVD), but worsening as noise increases. GEAP starts to lose accuracy slightly earlier.

empirically by solving systems of increasing degree d_Σ.

Figure 5a shows the per iteration computation time (time of an iteration of step 2 of Algorithm 1), verifying the asymptotic complexity of $\mathcal{O}(d_\Sigma^3)$. We see that this asymptotic behavior takes over at around degree 70. In total $r(d)(= d_\Sigma^2 - d_\Sigma)$ iterations are needed, leading to an asymptotic complexity of $\mathcal{O}(d_\Sigma^5)$.

In Figure 5b, the total time of the algorithm is shown for increasing degrees as well. Due to practical limitations, we can only show up to $d_\Sigma = 150$. As the asymptotic behaviour starts around 70, this is a rather limited range to show the



(a) Per iteration computation cost. (b) End-to-end computation cost.

Fig. 5: Per iteration (a) and end-to-end (b) computation cost. The measurements are the median of an adapted number of runs after warm-up such that the measurement of each point took at least five seconds. The per iteration cost is for step 2 in Algorithm 1, while the end-to-end cost also includes the transformation to and from Cauchy-like form, which is thus Algorithm 3. These costs are asymptotically $\mathcal{O}(d_\Sigma^3)$ and $\mathcal{O}(d_\Sigma^5)$ respectively although the asymptotics are only dominant after $d_\Sigma = 70$. The SVD operates at a cost of $\mathcal{O}(d_\Sigma^6)$.

complexity. An interesting observation is that our algorithm starts to perform faster than SVD from degree $d_\Sigma = 35$ onwards.

Not visible in these figures, but also important is memory consumption. The SVD stores the full matrix $\mathrm{M}(d)$ of size $\mathcal{O}(d_\Sigma^4)$, while our proposed method works directly on the compact Cauchy representation with size $\mathcal{O}(d_\Sigma^3)$. For illustration, the last point in Figure 5a, $d_\Sigma = 501$, which required $\sim$20GB would take a total computation time of $250500 \times 5.222\text{s} \approx 15$ days with our method, compared to $\sim$3TB and $\sim$105 days required with SVD (determined through extrapolation).

5. Generalizations.

An important question to answer is to what extent the ideas presented in the previous sections generalize to polynomial systems expressed in other bases or to systems involving more than two indeterminates. While Subsection 5.1 provides a (partial) answer to the first question by outlining an analogous fast algorithm for systems expressed in the Chebyshev basis, Subsection 5.2 addresses the challenges that one faces when dealing with more than two variables.

5.1. A fast algorithm for bivariate Chebyshev systems.

The Macaulay matrix for a Chebyshev system is introduced in Subsection 5.1.1. The reduction to a joint-GEVD problem is described in Subsection 5.1.2. The low-displacement rank structure of the Chebyshev-Macaulay matrix and its (fast) conversion to a Cauchy-like matrix are addressed in Subsection 5.1.3.

5.1.1. Macaulay matrix for Chebyshev systems.

Let $\{T_k(x)\}_{k=0}^{\infty}$ with $T_{k+1}(x) = 2x \cdot T_k(x) - T_{k-1}(x)$ and $T_0(x) = 1$, $T_1(x) = x$, denote the Chebyshev basis terms. Suppose that the system Σ in (1.1) is expressed with respect to this basis, i.e.,

$$\Sigma : \begin{cases} p_1(x,y) & := \displaystyle\sum_{i=0}^{d_\Sigma}\sum_{j=0}^{d_\Sigma - i} b_{1ij} T_i(x) T_j(y) = 0 \\ & \vdots \\ p_S(x,y) & := \displaystyle\sum_{i=0}^{d_\Sigma}\sum_{j=0}^{d_\Sigma - i} b_{Sij} T_i(x) T_j(y) = 0 \end{cases}. \tag{5.1}$$

In this setting, the columns of the Macaulay matrix correspond with the basis terms $\{T_i(x)T_j(y)\}_{i,j\geqslant 0,\, i+j\leqslant d}$, while the rows relate to the shifted polynomials

$$\{T_i(x)T_j(y)\cdot p_1, \ldots, T_i(x)T_j(y)\cdot p_S\}_{i,j\geqslant 0,\, i+j\leqslant \Delta d}.$$

Since $T_k(x)T_l(x) = \frac{1}{2}(T_{k+l}(x) + T_{|k-l|}(x))$, the entries of the Chebyshev-Macaulay matrix $\mathrm{W}(d) \in \mathbb{C}^{m(d)\times n(d)}$ will differ structurally from those of the Macaulay matrix associated with the monomial basis. In particular, if the entries are ordered in a non-graded lexicographically way (i.e., $T_{i_1}(x)T_{j_1}(y) < T_{i_2}(x)T_{j_2}(y)$ if $j_1 < j_2$, and in case $j_1 = j_2$, $i_1 < i_2$), the Chebyshev-Macaulay matrix will be, before the removal of certain rows and columns, a proper sum of a Toeplitz block-(block-)Toeplitz matrix with a Hankel block-(block-)Hankel matrix. For comparison, the Macaulay matrix for the monomial system involved only a Toeplitz term; see (2.5). Furthermore, this Toeplitz term had a upper-triangular structure, which is no longer the case for the Chebyshev system.

To construct the matrix $\mathrm{W}(d)$, we proceed as follows. For convenience, denote $\boldsymbol{b}_{kl} := \begin{bmatrix} b_{1kl} & \cdots & b_{Skl}\end{bmatrix}^\top$ for $k \leqslant d_\Sigma - l$ and $\boldsymbol{b}_{kl} = \mathbb{0}_S$, otherwise. Define $\mathrm{W}_j^{\mathrm{tpz}}, \mathrm{W}_j^{\mathrm{hnk}} \in \mathbb{C}^{S(\Delta d+1)\times d}$ as[14]

$$\mathrm{W}_j^{\mathrm{tpz}} := \begin{bmatrix} \boldsymbol{b}_{0j} & \boldsymbol{b}_{1j} & \cdots & \boldsymbol{b}_{d_\Sigma j} & & & & & \\ \boldsymbol{b}_{1j} & \boldsymbol{b}_{0j} & \boldsymbol{b}_{1j} & \cdots & \boldsymbol{b}_{d_\Sigma j} & & & & \\ \vdots & \ddots & \ddots & \ddots & & \ddots & & & \\ \boldsymbol{b}_{d_\Sigma j} & \cdots & \boldsymbol{b}_{1j} & \boldsymbol{b}_{0j} & \boldsymbol{b}_{1j} & \cdots & \boldsymbol{b}_{d_\Sigma j} & & \\ & \ddots & & \ddots & \ddots & \ddots & & \ddots & \\ & & \boldsymbol{b}_{d_\Sigma j} & \cdots & \boldsymbol{b}_{1j} & \boldsymbol{b}_{0j} & \boldsymbol{b}_{1j} & \cdots & \boldsymbol{b}_{d_\Sigma j} \end{bmatrix}, \tag{5.2a}$$

$$\mathrm{W}_j^{\mathrm{hnk}} := \begin{bmatrix} \boldsymbol{b}_{0j} & \boldsymbol{b}_{1j} & \cdots & \boldsymbol{b}_{d_\Sigma j} \\ \boldsymbol{b}_{1j} & & ⋰ & \\ \vdots & ⋰ & & \\ \boldsymbol{b}_{d_\Sigma j} & & & \\ & & & \end{bmatrix} \tag{5.2b}$$

for $j = 0, 1, \ldots, d_\Sigma$, respectively. Then, for $d \geqslant d_\Sigma$, the Macaulay matrix associated with the polynomial system (1.1) is given by

$$\mathrm{W}(d) := \tfrac{1}{2}\operatorname{diag}\{\mathrm{I}_{i,\Delta d+1}\otimes \mathrm{I}_S\}_{i=\Delta d+1}^{1}\left(\mathrm{W}^{\mathrm{tpz}}(d) + \mathrm{W}^{\mathrm{hnk}}(d)\right)\operatorname{diag}\{\mathrm{I}_{d+1,j}\}_{j=d+1}^{1}, \tag{5.3}$$

[14]In the presentation of (5.2a) and (5.2b), it is implicitly assumed that $\Delta d > d_\Sigma$ to reveal the full structure of the matrices.

where

$$\mathrm{W}^{\mathrm{tpz}}(d) := \begin{bmatrix} \mathrm{W}_0^{\mathrm{tpz}} & \mathrm{W}_1^{\mathrm{tpz}} & \cdots & \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & & & & & \\ \mathrm{W}_1^{\mathrm{tpz}} & \mathrm{W}_0^{\mathrm{tpz}} & \mathrm{W}_1^{\mathrm{tpz}} & \cdots & \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & & & & \\ \vdots & \ddots & \ddots & \ddots & & \ddots & & & \\ \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & \cdots & \mathrm{W}_1^{\mathrm{tpz}} & \mathrm{W}_0^{\mathrm{tpz}} & \mathrm{W}_1^{\mathrm{tpz}} & \cdots & \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & & \\ & \ddots & & \ddots & \ddots & \ddots & & \ddots & \\ & & \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & \cdots & \mathrm{W}_1^{\mathrm{tpz}} & \mathrm{W}_0^{\mathrm{tpz}} & \mathrm{W}_1^{\mathrm{tpz}} & \cdots & \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} \end{bmatrix},$$

$$\mathrm{W}^{\mathrm{hnk}}(d) := \begin{bmatrix} \mathrm{W}_0^{\mathrm{tpz}} & \mathrm{W}_1^{\mathrm{tpz}} & \cdots & \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & & \\ \mathrm{W}_1^{\mathrm{tpz}} & & \iddots & & & \\ \vdots & \iddots & & & & \\ \mathrm{W}_{d_\Sigma}^{\mathrm{tpz}} & & & & & \\ & & & & & \end{bmatrix}.$$

5.1.2. Joint-GEVD problem for Chebyshev systems. Starting with a column basis $\mathrm{P}(d)$ for null $W(d)$, the reduction of the root-solving problem (5.1) to a joint-GEVD problem is done in a similar way as done for the monomial-based system (1.1). Define $\mathbb{q}_d(t,x,y) \in \mathbb{C}^{n(d)}$ as

$$\mathbb{q}_d(t,x,y) := t^d \cdot \mathbb{q}_{d,x,y}(x/t, y/t), \tag{5.4}$$

where

$$\mathbb{q}_{d,x,y}(x,y) := \begin{bmatrix} \mathbb{q}_{d,x}^\top(x) & T_1(y) \cdot \mathbb{q}_{d-1,x}^\top(x) & \cdots & T_{d,x}(y) \cdot \mathbb{q}_0^\top(x) \end{bmatrix}^\top \in \mathbb{C}^{n(d)},$$

$$\mathbb{q}_{d,x}(x) := \begin{bmatrix} 1 & T_1(x) & \cdots & T_d(x) \end{bmatrix}^\top \in \mathbb{C}^{d+1}.$$

If $(t,x,y) \in \mathbb{P}^2(\mathbb{C})$ is a common root of the homogenized system Σ_h, then $\mathbb{q}_d(t,x,y) \in$ null $\mathrm{W}(d)$. Subsequently, if the system Σ_h only contains simple roots, the columns of

$$\mathbb{Q}(d) = \begin{bmatrix} \mathbb{q}_d(t_1,x_1,y_1) & \cdots & \mathbb{q}_d(t_{d_\Sigma^2}, x_{d_\Sigma^2}, y_{d_\Sigma^2}) \end{bmatrix} \in \mathbb{C}^{n(d)\times d_\Sigma^2} \tag{5.5}$$

will span null $\mathrm{W}(d)$ for $d \geqslant d^*$. Since $x \cdot T_0(x) = T_1(x)$, and $x \cdot T_k(x) = \frac{1}{2}(T_{k+1}(x) + T_{k-1}(x))$ for $k \geqslant 1$, the corresponding shift-matrices $\mathrm{K}_h(d) \in \mathbb{R}^{n(d-1)\times n(d)}$, for which the property $\mathrm{K}_h(d+1)\mathbb{q}_{d+1}(t,x,y) = h \cdot \mathbb{q}_d(t,x,y)$ hold for $h = \{t,x,y\}$, take on the form

$$\mathrm{K}_t(d) = \operatorname{diag}\{\mathrm{K}_{t,d-i}\}_{i=0}^{d}, \quad \mathrm{K}_{t,i} = \begin{bmatrix} 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix} \in \mathbb{C}^{i\times(i+1)},$$

$$\mathrm{K}_x(d) = \operatorname{diag}\{\mathrm{K}_{x,d-i}\}_{i=0}^{d}, \quad \mathrm{K}_{x,i} = \begin{bmatrix} 0 & 1 & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \in \mathbb{C}^{i\times(i+1)},$$

and

$$\mathrm{K}_y(d) = \begin{bmatrix} \mathbb{0}_{d\times(d+1)} & \mathrm{I}_d & & & \\ \frac{1}{2}\mathrm{I}_{d-1,d+1} & \mathbb{0}_{(d-1)\times d} & \frac{1}{2}\mathrm{I}_{d-1} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2}\mathrm{I}_{1,3} & \mathbb{0}_{1\times 2} & \frac{1}{2} \end{bmatrix}.$$

For $d \geqslant d^*$, this yields the joint-GEVD problem

$$\mathrm{L}_1\mathrm{A} = \mathbb{Q}(d)\mathrm{D}_t, \quad \mathrm{L}_2\mathrm{A} = \mathbb{Q}(d)\mathrm{D}_x, \quad \mathrm{L}_3\mathrm{A} = \mathbb{Q}(d)\mathrm{D}_y. \tag{5.6}$$

where $\mathrm{D}_x, \mathrm{D}_y, \mathrm{D}_t$ refer to the same matrices as in (2.13), $\mathrm{A} \in \mathbb{C}^{d_\Sigma^2 \times d_\Sigma^2}$ is an invertible matrix that satisfies $\mathrm{N}(d)\mathrm{A} = \mathbb{Q}(d)$, and $\mathrm{L}_i \in \mathbb{C}^{n(d)\times d_\Sigma^2}$ are given by

$$\mathrm{L}_1 := \mathrm{K}_t(d+1)\mathrm{P}(d+1), \quad \mathrm{L}_2 := \mathrm{K}_x(d+1)\mathrm{P}(d+1), \quad \mathrm{L}_3 := \mathrm{K}_y(d+1)\mathrm{P}(d+1).$$

5.1.3. Fast Cauchy conversion for Chebyshev-Macaulay matrices. Define

$$\mathrm{Y}_{p,\delta} := \begin{bmatrix} \delta & 1 & & & \\ 1 & 0 & \ddots & & \\ & 1 & \ddots & 1 & \\ & & \ddots & 0 & 1 \\ & & & 1 & \delta \end{bmatrix} \in \mathbb{C}^{p\times p}, \tag{5.7}$$

and let $\mathscr{D}_{\mathrm{cheb}} : \mathbb{C}^{m(d)\times n(d)} \to \mathbb{C}^{m(d)\times n(d)}$ be the operator

$$\mathscr{D}_{\mathrm{cheb}} : \quad \mathrm{X} \mapsto \mathrm{diag}\left\{\mathrm{Y}_{i,0} \otimes \mathrm{I}_S\right\}_{i=\Delta d+1}^{1} \mathrm{X} - \mathrm{X}\,\mathrm{diag}\left\{\mathrm{Y}_{j,\delta_j}\right\}_{j=d+1}^{1}, \tag{5.8}$$

for some choice of $\{\delta_j\}_{j=1}^{d+1} \subset (0,1]$. A counting argument would reveal that the displacement rank of (5.3) with respect to (5.8) is bounded by

$$\mathrm{rank}\,\mathscr{D}_{cheb}\left\{\mathrm{W}(d)\right\} \leqslant 2(d+1) + 2S(\Delta d+1) = 2\left((S+1)(d+1) - Sd_\Sigma\right), \tag{5.9}$$

which reveals that the rank grows at the same pace as for the monomial case (see (3.5)), but with slightly larger constants.

Remark 5.1. Since $\mathrm{M}(d) = \mathrm{E}\mathrm{W}(d)\mathrm{J}$ for some invertible E and J, note that it is always possible to define a displacement operator for which the displacement rank of $\mathrm{W}(d)$ equals that of (3.5). However, this implicitly involves converting the Chebyshev system into a monomial system; a potentially highly ill-conditioned operation.

Furthermore, (5.7) is known to have a "fast" eigendecomposition. Indeed, if $\delta = 0$ and $\delta = 1$, the eigendecompositions are respectively

$$\mathrm{Y}_{p,0} = \mathrm{S}_p\,\mathrm{diag}\left\{2\cos\left(\tfrac{j\pi}{p+1}\right)\right\}_{j=1}^{p} \mathrm{S}_p^\top, \quad \mathrm{Y}_{p,1} = \mathrm{C}_p\,\mathrm{diag}\left\{2\cos\left(\tfrac{(j-1)\pi}{p}\right)\right\}_{j=1}^{p} \mathrm{C}_p^\top,$$

where $[\mathrm{S}_p]_{ij} := \sqrt{\frac{2}{p+1}}\sin\frac{ij\pi}{p+1}$, $[\mathrm{C}_p]_{ij} := \sqrt{\frac{2}{p}}\kappa_j\cos\left(\frac{(2i+1)(j-1)\pi}{2p}\right)$, $\kappa_j = \frac{1}{\sqrt{2}}$ for $j = 1$ and $k_j = 0$ otherwise [8,23]. The matrix S_p (C_p) is the discrete sine (cosine) transform and has a fast matrix-vector multiply; see e.g., [24, Section 1.4.2]. For $0 < \delta < 1$, the eigenvalues interlace between those of $\mathrm{Y}_{p,0}$ and $\mathrm{Y}_{p,1}$. Since $\mathrm{Y}_{p,\delta}$ is a rank-two update of $\mathrm{Y}_{p,0}$ (or $\mathrm{Y}_{p,1}$), the eigenmatrix of $\mathrm{Y}_{p,\delta}$ can further be expressed as the product of S_p (or C_p) with a Cauchy-like matrix, whose matrix-vector product can also be efficiently evaluated using the fast multipole method [25,27]. Similar to the ϕ_j's in (3.2), $\{\delta_j\}_{j=1}^{d+1}$ can be chosen to put the eigenvalues of $\mathrm{diag}\left\{\mathrm{Y}_{j,\delta_j}\right\}_{j=d+1}^{1}$ at the desired locations, so that one can proceed in the same manner as for monomial case discussed in Section 3. The complexity of the algorithm is again $\mathcal{O}(d_\Sigma^5)$, but slightly larger constants will be involved.

5.2. Extending the technique for systems with more than two variables. For polynomial systems with more than two variables, the Macaulay matrix is multi-level Toeplitz, as opposed to the two-level Toeplitz structure for the bivariate case (2.5). With a similar strategy by applying displacement rank theory on the inner-most blocks, the law of diminishing returns applies and a complexity reduction from $\mathcal{O}(d^{3n})$ to $\mathcal{O}(d^{3n-1})$ is only achieved for a non-degenerate n-variable system. A full exploitation of the multi-level Toeplitz structure remains an open question.

6. Conclusions and future work.

We introduced a fast algorithm to compute a numerical basis for the right null space of the Macaulay matrix associated with a bivariate polynomial system. The algorithm applies displacement rank theory to the inner Toeplitz blocks of the Macaulay matrix to convert it into a Cauchy form so that subsequently the null space can be determined efficiently from a rank-revealing LU-factorization. Initial numerical experiments show that the algorithm is stable. Furthermore, a similar fast algorithm was also outlined for polynomial systems expressed in the Chebyshev basis.

This work has raised several open questions. Firstly, the search for better pivoting strategies is something worth pursuing. Secondly, it is noted that the current method relies on the exact algebraic properties of Cauchy-like matrices to allow for fast Gaussian elimination. The question arises whether the approximate low-rank properties of Cauchy-like matrices [35] can be exploited to design even faster algorithms. We conjecture that the complexity can be further reduced from $\mathcal{O}(d_\Sigma^5)$ to $\mathcal{O}(d_\Sigma^4 \log^p d_\Sigma)$ (for some $p > 1$) by using the techniques proposed in [15, 44]. Thirdly, it is not exactly clear how the presented method exactly fits into the framework of the "degree-by-degree" recursive algorithm from [4, 41]. In practice, the degree of regularity is often attained well before the bound (2.11), so that incremental methods of building the null space can lead to significant savings. Therefore, it is worth investigating whether the recursive and Toeplitz properties of the Macaulay matrix can somehow be simultaneously exploited. Fourthly, a refinement algorithm could be designed to mitigate the loss of accuracy while maintaining the asymptotic complexity. Lastly, it is still unclear how to fully exploit multi-level Toeplitz structures should in the general n-variable case.

Acknowledgements. The authors would like to thank (i) Matías R. Bender and Simon Telen for their useful discussions and pointers during the CWI workshop in Amsterdam on solving polynomial systems, (ii) Philippe Dreesen for his email correspondence to clarify certain theoretical aspects behind Macaulay matrices.

REFERENCES

[1] 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. 1

[2] D. J. Bates, A. J. Sommese, J. D. Hauenstein, and C. W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, SIAM, Philadelphia, PA, USA, 2013. 1

[3] K. Batselier, *A Numerical Linear Algebra Framework for Solving Problems with Multivariate Polynomials*, PhD thesis, KU Leuven, Leuven, Belgium, 2013. 1, 2, 2.3

[4] 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. 1, 2.3, 6

[5] M. R. Bender and S. Telen, *Toric eigenvalue methods for solving sparse polynomial systems*, Math. Comput., 91 (2022), pp. 2397–2429. 1

[6] D. A. Bini and P. Boito, *A fast algorithm for approximate polynomial GCD based on struc-*

tured matrix computations, in Numerical Methods for Structured Matrices and Applications, D. Bini, V. Mehrmann, V. Olshevsky, E. Tyrtyshnikov, and M. Van Barel, eds., vol. 199 of Operator Theory: Advances and Applications, Birkhäuser, Basel, Switzerland, 2010. 1.3

[7] D. A. Bini and A. Marco, *Computing curve intersection by means of simultaneous iterations*, Numerical Algorithms, 43 (2006), pp. 151–175. 1.3

[8] E. Bozzo and C. Di Fiore, *On the use of certain matrix algebras associated with discrete trigonometric transforms in matrix displacement decomposition*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 312–326. 5.1.3

[9] B. Buchberger, *An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal (Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal)*, PhD thesis, University of Innsbruck, Innsbruck, Austria, 1965. 1

[10] S. Chandrasekaran, N. Govindarajan, and A. Rajagopal, *Fast algorithms for displacement and low-rank structured matrices*, in Proc. of the 2018 ACM International Symposium on Symbolic and Algebraic Computation, ISSAC '18, New York, NY, USA, 2018, Association for Computing Machinery, p. 17–22. 1, 3.1

[11] 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. 2.3

[12] D. A. Cox, *Stickelberger and the eigenvalue theorem*, in Commutative Algebra, Springer, 2021, pp. 283–298. 1

[13] D. A. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer Science & Business Media, New York, NY, USA, 2013. 1

[14] D. A. Cox, J. B. Little, and D. O'Shea, *Using Algebraic Geometry*, vol. 185 of Graduate Texts in Mathematics, Springer, 1998. 1, 2.3

[15] P. Dewilde and A.-J. Van der Veen, *Time-Varying Systems and Computations*, Springer Science & Business Media, Dordrecht, The Netherlands, 1998. 1, 6

[16] P. Dreesen, *Back to the Roots: Polynomial System Solving Using Linear algebra*, PhD thesis, KU Leuven, Leuven, Belgium, 2013. 1, 2, 2.3

[17] P. Dreesen, K. Batselier, and B. De Moor, *Multidimensional realisation theory and polynomial system solving*, Internat. J. Control, 91 (2018), pp. 2692–2704. 1, 2, 2.3

[18] C. Eder and J.-C. Faugère, *A survey on signature-based algorithms for computing Gröbner bases*, J. Symb. Comput., 80 (2017), pp. 719–784. 1

[19] I. Z. Emiris and B. Mourrain, *Matrices in elimination theory*, J. Symb. Comput., 28 (1999), pp. 3–44. 1

[20] I. Z. Emiris and V. Y. Pan, *Symbolic and numeric methods for exploiting structure in constructing resultant matrices*, J. Symb. Comput., 33 (2002), pp. 393–413. 1

[21] E. Evert, M. Vandecappelle, and L. De Lathauwer, *A recursive eigenspace computation for the canonical polyadic decomposition*, SIAM J. Matrix Anal. Appl., 43 (2022), pp. 274–300. 1, 2.3, 2.3

[22] L. Giraud, J. Langou, M. Rozložník, and J. v. d. Eshof, *Rounding error analysis of the classical Gram-Schmidt orthogonalization process*, Numer. Math., 101 (2005), pp. 87–100. 3.3

[23] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comput., 64 (1995), pp. 1557–1576. 5.1.3

[24] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, USA, 4 ed., 2012. 3.2.3, 3.3, 5.1.3

[25] L. Greengard and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comput. Phys., 73 (1987), pp. 325–348. 5.1.3

[26] M. Gu, *Stable and efficient algorithms for structured systems of linear equations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 279–306. 1.2, 3.2.4, 3.2.4

[27] M. Gu and S. C. Eisenstat, *A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276. 5.1.3

[28] R. Hartshorne, *Algebraic Geometry*, vol. 52, Springer Science & Business Media, New York, NY, USA, 2013. 2.2

[29] G. Heinig, *Inversion of generalized Cauchy matrices and other classes of structured matrices*, in Linear Algebra for Signal Processing, A. Bojanczyk and G. Cybenko, eds., vol. 69 of The IMA Volumes in Mathematics and its Applications, Springer, New York, NY, USA, 1995. 1.2, 3.2.3

[30] T. Kailath, S.-Y. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979), pp. 395–407. 1

[31] T. Kailath and A. H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37 (1995), pp. 297–386. 1, 3.1

[32] D. lazard, *Résolution des systèmes d'équations algébriques*, Theoret. Comput. Sci., 15 (1981), pp. 77–110. 1

[33] T. Y. Li, *Numerical solution of multivariate polynomial systems by homotopy continuation methods*, Acta Numer., 6 (1997), p. 399–436. 1

[34] F. S. Macaulay, *Some formulae in elimination*, Proc. London Math. Soc, 1 (1902), pp. 3–27. 1, 2.3

[35] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A fast algorithm for the inversion of general toeplitz matrices*, Comput. Math. Appl., 50 (2005), pp. 741–752. 6

[36] N. Mastronardi, M. Van Barel, and R. Vandebril, *On the computation of the null space of Toeplitz-like matrices*, Electron. Trans. Numer. Anal., 33 (2009), pp. 151–162. 1.3

[37] L. Miranian and M. Gu, *Strong rank revealing LU factorizations*, Linear Algebra Appl., 367 (2003), pp. 1–16. 1.2, 3.2, 3.2.1, 3.2.1, 3.2.4

[38] 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. 1

[39] B. Mourrain and V. Y. Pan, *Asymptotic acceleration of solving multivariate polynomial systems of equations*, in Proc. Annu. ACM Symp. Theory Comput., STOC '98, New York, NY, USA, 1998, Association for Computing Machinery, p. 488–496. 1.3

[40] B. Mourrain and V. Y. Pan, *Multivariate polynomials, duality, and structured matrices*, Journal of complexity, 16 (2000), pp. 110–180. 1.3

[41] 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. 1, 1.2, 2.3, 12, 6

[42] Y. Nakatsukasa and V. Noferini, *On the stability of computing polynomial roots via confederate linearizations*, Math. Comput., 85 (2016), pp. 2391–2425. 1.2

[43] V. Noferini and J. Pérez, *Chebyshev rootfinding via computing eigenvalues of colleague matrices: when is it stable?*, Math. Comput., 86 (2017), pp. 1741–1767. 1.2

[44] X. Ou and J. Xia, *SuperDC: Superfast divide-and-conquer eigenvalue decomposition with improved stability for rank-structured matrices*, SIAM J. Sci. Comput., 44 (2022), pp. A3041–A3066. 6

[45] C.-T. Pan, *On the existence and computation of rank-revealing LU factorizations*, Linear Algebra Appl., 316 (2000), pp. 199–222. 1.2, 3.2, 3.2.4

[46] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, 2005. 1, 2.2

[47] L. Sorber, M. Van Barel, and L. De Lathauwer, *Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization*, SIAM J. Optim., 23 (2013), pp. 695–720. 2.3

[48] H. J. Stetter, *Matrix eigenproblems are at the heart of polynomial system solving*, SIGSAM Bull., 30 (1996), p. 22–25. 1

[49] H. J. Stetter, *Numerical polynomial algebra*, SIAM, Philadelphia, PA, USA, 2004. 1

[50] B. Sturmfels, *Solving systems of polynomial equations*, no. 97, American Mathematical Soc., 2002. 2.2

[51] 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. 1

[52] J. Vanderstukken and L. De Lathauwer, *Systems of polynomial equations, higher-order rensor decompositions and multidimensional harmonic retrieval: A unifying framework. Part I: The canonical polyadic decomposition*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 883–912. 1, 2, 2.3

[53] J. Vanderstukken, P. Kurschner, 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. 2.3

[54] J. Verschelde, *Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation*, ACM Trans. Math. Softw., 25 (1999), p. 251–276. 1

[55] J. Verschelde, P. Verlinden, and R. Cools, *Homotopies exploiting Newton polytopes for solving sparse polynomial systems*, SIAM J. Numer. Anal., 31 (1994), pp. 915–930. 1

[56] N. Vervliet and L. De Lathauwer, *Numerical optimization-based algorithms for data fusion*, in Data Handling in Science and Technology, vol. 31, Elsevier, 2019, pp. 81–128. 2.3

[57] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410. 1