Efficient Computation of Macaulay Matrix Null Spaces Through Exploiting Shift-Invariant Structures

Nithin Govindarajan

with Raphaël Widdershoven, Shiv Chandrasekaran, and Lieven De Lathauwer

June 15th 2023

KU LEUVEN

Overview

The problem that we wish to solve

Let $S \geq N$, and consider system of the multivariate polynomials

$$\Sigma : \quad \begin{cases} p_1 &= p_1(x_1, x_2, \ldots, x_N) \\ & \vdots \\ p_S &= p_S(x_1, x_2, \ldots, x_N) \end{cases}, \tag{1}$$

with $\deg(p_s) = d_s$.

Goal:

Find all roots $\left\{(t^{(r)}, x_1^{(r)}, \ldots, x_N^{(r)})\right\}_{r=1}^{R}$ of the *homogenized* system Σ_h.

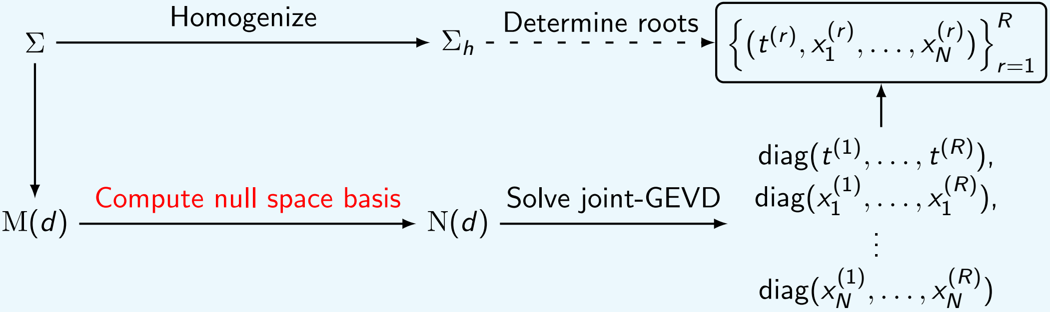
NOTE: In practice, we may be only interested in the affine roots...

The Macaulay matrix $\mathrm{M}(d)$ and its right null space

The rows of $\mathrm{M}(d)$ span the set of polynomial combinations

$$\left\{ \sum_{s=1}^{S} h_s \cdot p_s : \quad \deg(h_s) = d - d_s \right\}.$$

Null space computation is a major computational bottleneck in *many* algorithms!



(Vanderstukken and De Lathauwer 2021)

Overview

Focus: (Possibly overdetermined) bivariate polynomial systems

For simplicity of exposition, we assume $\deg(p_s) = d_\Sigma$, i.e.,

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

In lex ordering Macaulay is almost Toeplitz block-(block-)Toeplitz!

$M(4) =$

	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 for the general bivariate case

Let $\Delta d := d - d_\Sigma$. Then,

$$\mathrm{M}(d) := \begin{bmatrix} \mathrm{M}_{0,0} & \mathrm{M}_{1,0} & \cdots & \mathrm{M}_{d_\Sigma,0} & & & \\ & \mathrm{M}_{0,1} & \mathrm{M}_{1,1} & \cdots & \mathrm{M}_{d_\Sigma,1} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & \mathrm{M}_{0,\Delta d} & \mathrm{M}_{1,\Delta d} & \cdots & \mathrm{M}_{d_\Sigma,\Delta d} \end{bmatrix} \in \mathbb{C}^{\frac{S}{2}(\Delta d+1)(\Delta d+2)\times\frac{1}{2}(d+1)(d+2)},$$

with

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

Intermezzo: The generalized shift matrix

Let

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

Eigen-decomposition of $\mathrm{Z}_{p,\varphi}$

Denote $\omega_p := \exp(-2\pi\iota/p)$ and $\mathrm{F}_p \in \mathbb{C}^{p\times p}$ the (unitary) DFT matrix. Then

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

where $\mathrm{D}_{p,\varphi} := \mathrm{diag}(1, \varphi^{-1/p}, \ldots, \varphi^{-(p-1)/p})$, $\Omega_p := \mathrm{diag}(1, \bar{\omega}_p, \ldots, \bar{\omega}_p^{p-1})$,.

Toeplitz matrices and their low "displacement rank" properties

Consider the so-called displacement equation

$$
\mathrm{Z}_{4,1}\begin{bmatrix} t_0 & t_1 & t_2 & t_3 \\ t_{-1} & t_0 & t_1 & t_2 \\ t_{-2} & t_{-1} & t_0 & t_1 \\ t_{-3} & t_{-2} & t_{-1} & t_0 \end{bmatrix} - \begin{bmatrix} t_0 & t_1 & t_2 & t_3 \\ t_{-1} & t_0 & t_1 & t_2 \\ t_{-2} & t_{-1} & t_0 & t_1 \\ t_{-3} & t_{-2} & t_{-1} & t_0 \end{bmatrix}\mathrm{Z}_{4,\phi}
$$

$$
=
$$

$$
\underbrace{\begin{bmatrix} t_{-3}-t_1 & t_{-2}-t_2 & t_{-1}-t_3 & t_0-\varphi t_0 \\ 0 & 0 & 0 & t_3-\varphi t_{-1} \\ 0 & 0 & 0 & t_2-\varphi t_{-2} \\ 0 & 0 & 0 & t_1-\varphi t_{-3} \end{bmatrix}}_{\text{rank is only two!}}
$$

.

NOTE: $\varphi \in \mathbb{C}$ is chosen such that the operator $\mathscr{D} : \mathrm{T} \mapsto \mathrm{Z}_{4,1}\mathrm{T} - \mathrm{T}\mathrm{Z}_{4,\varphi}$ remains invertible!

The key observation that shall allow for a faster algorithm!

Consider the displacement operator

$$\mathscr{D}\left\{\mathrm{M}(d)\right\} = \begin{bmatrix} \mathrm{Z}_{d+1,1} \otimes \mathrm{I}_S & & \\ & \ddots & \\ & & \mathrm{Z}_{1,1} \otimes \mathrm{I}_S \end{bmatrix} \mathrm{M}(d) - \mathrm{M}(d) \begin{bmatrix} \mathrm{Z}_{d+1,\varphi_{d+1}} & & \\ & \ddots & \\ & & \mathrm{Z}_{1,\varphi_1} \end{bmatrix}.$$

$\mathrm{M}(d)$ has relative "low" displacement rank too!

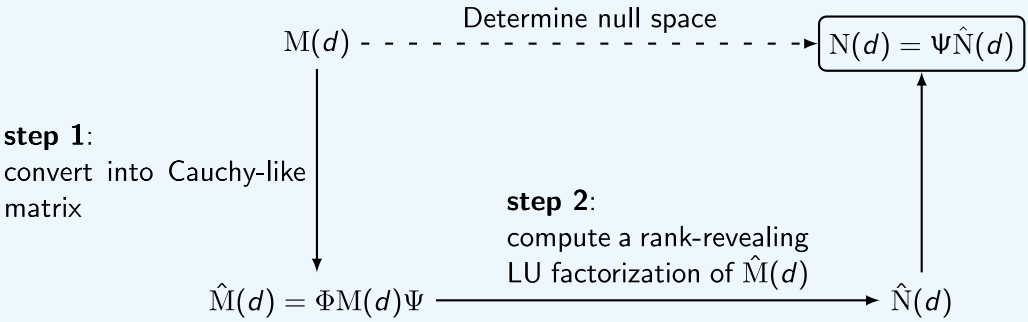
Dimensions of $\mathrm{M}(d) \in \mathbb{C}^{\frac{S}{2}(\Delta d+1)(\Delta d+2) \times \frac{1}{2}(d+1)(d+2)}$ grow *quadratically* w.r.t. d, but

$$\operatorname{rank} \mathscr{D}\left\{\mathrm{M}(d)\right\} \leq S(\Delta d + 1) = S\left(d + 1 - d_\Sigma\right).$$

grows only *linearly* with d.

Overview

An overview of the fast algorithm



Both steps can be done *fast*!

Step 1: convert into Cauchy-like matrix

Apply unitary transformations Φ and Ψ such that $\Phi \mathrm{M}(d) \Psi =: \hat{\mathrm{M}}(d)$ is Cauchy-like, i.e., its entries are of the form

$$\left[\hat{\mathrm{M}}(d)\right]_{ij} := [\Phi \mathrm{M}(d) \Psi]_{ij} = \frac{\boldsymbol{u}_i^* \boldsymbol{v}_j}{\mu_i - \nu_j}, \qquad \boldsymbol{u}_i, \boldsymbol{v}_j \in \mathbb{C}^{S(\Delta d+1)}.$$

Step 1 *done fast*: use displacement rank theory!

Best explained through the simpler Toeplitz case...

$$\mathrm{Z}_{n,1}\mathrm{T}_n - \mathrm{T}_n\mathrm{Z}_{n,\varphi} = \mathrm{U}\mathrm{V}^*$$

$$\downarrow$$

$$(F_n\Omega_n\mathrm{F}_n^*)\,\mathrm{T}_n - \mathrm{T}_n\left((\mathrm{D}_{n,\varphi}F_n)(\varphi^{1/n}\Omega_n)(\mathrm{D}_{n,\varphi}\mathrm{F}_n)^{-1}\right) = \mathrm{U}\mathrm{V}^*$$

$$\downarrow$$

$$\Omega_n\mathrm{F}_n^*\mathrm{T}_n\mathrm{D}_{n,\varphi}\mathrm{F}_n - \mathrm{F}_n^*\mathrm{T}_n\mathrm{D}_{n,\varphi}\mathrm{F}_n\left(\varphi^{1/n}\Omega_n\right) = (F_n^*\mathrm{U})\mathrm{F}_n^*\mathrm{D}_{n,\varphi}^*\mathrm{V}^*$$

$$\downarrow$$

$$\mathrm{diag}(\boldsymbol{\mu})\mathrm{C} - \mathrm{C}\mathrm{diag}(\boldsymbol{\nu}) = \mathrm{R}\mathrm{S}^* =: \mathrm{G}$$

$$\downarrow$$

Displacement equation for *Cauchy-like* matrix!

Step 2: compute a rank-revealing LU factorization of $\hat{\mathrm{M}}(d)$

Let $r(d) := \operatorname{rank} \mathrm{M}(d)$. Compute a rank-revealing LU (RRLU) factorization

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

Expression for the null space $\mathrm{N}(d)$

$$\mathrm{N}(d) = \Psi \Pi_2 \begin{bmatrix} \tilde{\mathrm{N}} \\ \mathrm{I}_{d_\Sigma^2} \end{bmatrix}, \qquad \tilde{\mathrm{N}} := -\hat{\mathrm{M}}_{11}^{-1}\hat{\mathrm{M}}_{12}.$$

NOTE: Gaussian elimination on Macaulay matrix $\equiv$ polynomial reductions for e.g., Grobner Basis reductions (Eder and Faugère 2017)

Step 2 *done fast*: Apply Schur algorithm on Cauchy-like matrix (Heinig 1995)

$$\mathsf{diag}(\boldsymbol{\mu}) \begin{bmatrix} \mathrm{A} & \mathrm{G}^* \\ \mathrm{F} & \mathrm{B} \end{bmatrix} - \begin{bmatrix} \mathrm{A} & \mathrm{G}^* \\ \mathrm{F} & \mathrm{B} \end{bmatrix} \mathsf{diag}(\boldsymbol{\nu}) = \begin{bmatrix} \mathrm{R}_1 \\ \mathrm{R}_2 \end{bmatrix} \begin{bmatrix} \mathrm{S}_1 \\ \mathrm{S}_2 \end{bmatrix}^*$$

$$\downarrow$$

$$\mathsf{diag}(\boldsymbol{\mu}) \begin{bmatrix} \mathrm{A} & \\ & \mathrm{B} - \mathrm{F}\mathrm{A}^{-1}\mathrm{G}^* \end{bmatrix} - \begin{bmatrix} \mathrm{A} & \\ & \mathrm{B} - \mathrm{F}\mathrm{A}^{-1}\mathrm{G}^* \end{bmatrix} \mathsf{diag}(\boldsymbol{\nu}) = \begin{bmatrix} \mathrm{R}_1 \\ \tilde{\mathrm{R}}_2 \end{bmatrix} \begin{bmatrix} \mathrm{S}_1 \\ \tilde{\mathrm{S}}_2 \end{bmatrix}^*,$$

where $\tilde{\mathrm{R}}_2 = \mathrm{R}_2 - \mathrm{F}\mathrm{A}^{-1}\mathrm{R}_1$ and $\tilde{\mathrm{S}}_2 = \mathrm{S}_2 - \mathrm{G}(\mathrm{A}^*)^{-1}\mathrm{S}_1$,

Main idea of Schur algorithm

Perform Gauss elimination on the *generators* instead of the dense matrix itself!

Step 2 *done fast*: *approximate* total pivoting through QR-decomposition of generators

Recall

$$\text{diag}(\boldsymbol{\mu})\mathrm{C} - \mathrm{C}\text{diag}(\boldsymbol{\nu}) = \mathrm{R}\mathrm{S}^* =: \mathrm{G} \in \mathbb{C}^{n\times n}.$$

(Gu 1998, Lemma 3.1)

Let j_{max} denote the column with largest 2-norm in G. Then,

$$\max_{1\le i\le n} |c_{ij_{\text{max}}}| \ge \frac{1}{K\sqrt{n}} \max_{1\le i,j\le n} |c_{ij}|,$$

with $K := \max_{1\le i,j,\imath,\jmath\le n} |\mu_i - \nu_j| / |\mu_\imath - \nu_\jmath|$.

Full QR-decomposition of R expensive! $\rightarrow$ Use *fancy* QR updating techniques!

Overview

We are able to reduce the flop complexity from $\mathcal{O}(d_\Sigma^6)$ to $\mathcal{O}(d_\Sigma^5)$!

Assumptions:

- Σ has a consistent set of equations $\rightarrow$ number of roots $= d_\Sigma^2$,
- $S \ll d_\Sigma$.

A quick complexity overview for each step

- Step 1: $\mathcal{O}(S \cdot d_\Sigma \cdot \Delta d \cdot d \log d)$
- Step 2: $\mathcal{O}(r(d) \cdot S^2 d^3)$

$$d \leq 2d_\Sigma - 2 \text{ to find a null space containing all system roots}$$
$$\downarrow$$
$$\mathcal{O}(d_\Sigma^5)$$

NOTE: We can overcome the restriction $S \ll d_\Sigma$ by random sampling of the rows!

Overview

The error metric used to assess performance

Let $\mathrm{Q} \in \mathbb{C}^{n(d) \times d_\Sigma^2}$ be an orthonormal basis for col $\mathrm{N}(d)$, then define

$$\epsilon := \frac{\|\mathrm{M}(d)\mathrm{Q}\|_2}{\|\mathrm{M}(d)\|_2} \geq \frac{\sigma_{r(d)+1}}{\sigma_1} =: \epsilon_{\mathsf{min}},$$

Algorithm stability: error grows linearly with problem size

Median error over 100 runs for *square* systems with different methods and degrees.

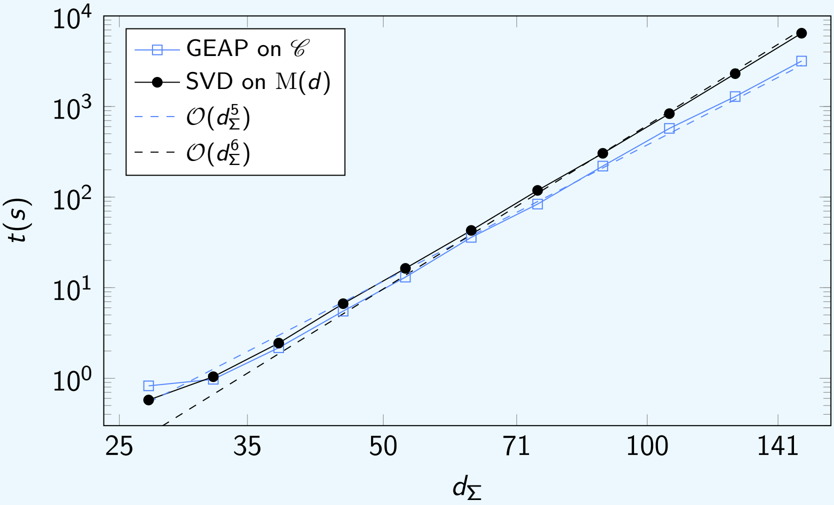
	d_Σ				
	2	4	8	16	32
SVD on $\mathrm{M}(d)$	2.23e-16	3.75e-16	5.70e-16	7.94e-16	9.51e-16
SVD on $\hat{\mathrm{M}}(d)$	2.57e-16	4.77e-16	7.54e-16	9.97e-16	1.15e-15
GECP on $\mathrm{M}(d)$	1.40e-16	3.11e-16	8.33e-16	1.02e-14	1.40e-13
GECP on $\hat{\mathrm{M}}(d)$	2.08e-16	4.65e-16	1.03e-15	9.73e-15	1.21e-13
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

Sources of error:

- switching to LU instead of an SVD
- working with the compact Cauchy representation $\mathscr{C}$
- switching to approximate pivoting $\leftarrow$ Surprisingly not so bad!

Our experiments indicate that the flop complexity is indeed $O(d_\Sigma^5)$

The measurements are the median of an adapted number of runs after warmup.



Overview

The algorithm generalizes to Chebyshev systems!

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

Key ideas to arrive to an $\mathcal{O}(d_\Sigma^5)$ algoritm:

- *Toeplitz-plus-Hankel* instead of just Toeplitz.
- Apply same techniques but with a *modified* displacement equation.

The algorithm does not nicely extend for general N-dimensional systems

- Displacement rank theory does not generalize nicely to higher dimensions ☹
- Diminishing returns: $\mathcal{O}(d^{3N})$ to $\mathcal{O}(d^{3N-1})$
- Open problem: how to exploit multi-level Toeplitz structures?

Overview

Conclusions

- An asymptotically *faster* algorithm for Macaulay null space computation.
- Generalizes to Chebyshev systems
- Generic n-dimensional systems still a challenge! (assume additional structure?)

To learn more, see our pre-print:

Govindarajan, Nithin et al. (2023). "A fast algorithm for computing Macaulay nullspaces of bivariate polynomial systems". In: Technical Report 23-16, ESAT-STADIUS, KU Leuven (Leuven, Belgium).

References I

Eder, Christian and Jean-Charles Faugère (2017). "A survey on signature-based algorithms for computing Gröbner bases". In: *J. Symb. Comput.* 80, pp. 719–784.

Gu, Ming (1998). "Stable and Efficient Algorithms for Structured Systems of Linear Equations". In: *SIAM J. Matrix Anal. Appl.* 19.2, pp. 279–306.

Heinig, Georg (1995). "Inversion of Generalized Cauchy Matrices and other Classes of Structured Matrices". In: *Linear Algebra for Signal Processing*. Ed. by A. Bojanczyk and G. Cybenko. Vol. 69. The IMA Volumes in Mathematics and its Applications. New York, NY, USA: Springer.

Vanderstukken, Jeroen and Lieven De Lathauwer (2021). "Systems of Polynomial Equations, Higher-Order Rensor Decompositions and Multidimensional Harmonic Retrieval: A Unifying Framework. Part I: The Canonical Polyadic Decomposition". eng. In: *SIAM J. Matrix Anal. Appl.* 42.2, pp. 883–912.

References II

Vanderstukken, Jeroen et al. (2021). "Systems of Polynomial Equations, Higher-Order Tensor Decompositions, and Multidimensional Harmonic Retrieval: A Unifying Framework. Part II: The Block Term Decomposition". In: *SIAM J. Matrix Anal. Appl.* 42.2, pp. 913–953.