KU LEUVEN

STADIUS
Center for Dynamical Systems, Signal Processing and Data Analytics

Citation/Reference	Authors: Lukas Vanpoucke, Bart De Moor Title: Direct computation of Multiplication maps Journal: Vol: Pages:
Archived version	■ To be submitted ☐ Submitted version ☐ Accepted (Author manuscript: the content is identical to the content of the published paper, but without the final typesetting by the publisher) ☐ Published ☐ Published (Open Access)
Acknowledgements	In this manuscript, the following funding is acknowledged: ☐ EU H22020 ■ EU Horizon Europe ■ FWO ■ KULeuven Internal Funds ■ VLAIO ☐ AI ■ ERC ■ Other
Published version	
Copyright statement	
Author contact	lukas.vanpoucke@esat.kuleuven.be



Direct construction of multiplication maps

Lukas Vanpoucke, Bart De Moor *

Abstract

This work presents a novel algorithm to compute multiplication maps for numerical multivariate polynomial root finding. The algorithm relies on insights from symbolic root finding methods to reduce the size of intermediate matrices compared to common numerical approaches. Contrary to the symbolic approaches, the presented method is agnostic about the underlying polynomials that are being represented, rather it only relies on the coefficient matrices. To this end, a rank-based check is introduced to decide upon the number of underlying basis elements to add in each iteration, leading to a linear algebra based approach. While a full-blown proof of convergence remains an open research question, brief numerical experiments are provided to showcase the efficacy of the method.

1 Introduction

The current work proposes an novel algorithm for numerical multivariate polynomial root finding. For this class of problems, there is a broad variety of techniques that can roughly be subdivided in two classes: homotopy-based methods [3, 23] and algebraic methods. We will focus on the latter class of approaches, which most often rely on a set of joint eigenvalue problems to recover the common roots using linear algebra-techniques [17]. Within this subclass of techniques, there still exists a multitude of different approaches, a brief overview of which is presented in the following section, each with their own strengths and weaknesses [18].

The current work presents a novel algebraic root finding approach for polynomial problems with a finite number of projective roots. The proposed method aims at aims at combining insights from both border basis-style algorithms [17, 10, 14, 21, 12, 16] and resultant-based methods [1, 7, 19, 20, 11, 9, 8]. However,

*This work was supported in part by the KU Leuven Research Fund (projects iBOF/24/064 and IDN/24/013), the Industrial Research Fund and several LRD bilateral industrial projects; in part by the Flemish Government agencies: FWO (TBM T001725N), EWI through the Flanders AI Research Program and VLAIO (CSBO - HBC.2021.0076); and in part by the European Commission (ERC Adv. Grant) under Grant 885682. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or ERC. Neither the European Union nor the granting authority can be held responsible for them.

contrary to these methods, this approach relies on a rank decision to decide how many basis polynomials should be added at each iteration. This property essentially eliminates any bookkeeping about the underlying polynomials, resulting in a flexible algorithm that solely acts on the coefficient matrices of these implicit polynomials through the well-established techniques of numerical linear algebra.

First a brief overview of algebraic numerical polynomial root finding techniques is presented in Section 2, after which the main iteration and the initialization and convergence of the novel method is discussed in Sections 3 and 4. Section 5 then presents some numerical experiments to further investigate the performance of the algorithm. Finally, an overall conclusion is drawn in in Section 6 and future challenges are identified.

2 Eigenvalue-based root finding

This section elaborates on the basics underpinning eigenvalue-based polynomial root finding, that is, multiplication matrices and the reduction of onto a polynomial by a set of polynomials. Let us start with a an example based on the border basis approach to illustrate the general idea of such methods.

Example 2.1 *Consider the following set of polynomial equations* $f(x,y) = 1 + x + y = 0$; $g(x,y) = 2 - 3x + x^2 = 0$. *New equations can then be derived through polynomial combinations of these equations, e.g.* $0 = g(x,y) - xf(x,y) = 2 - 4x - xy$. *These equations thus imply that the following relations hold between the monomials* $\{1, x, y, x^2, xy\}$ *in the common roots of* f *and* g*:*

$$\begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \end{bmatrix} = \begin{bmatrix} 1 & x \\ 1 & 0 \\ 0 & 1 \\ -1 & -1 \\ -2 & 3 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}. \tag{1}$$

Here, the monomials $\boldsymbol{b}(\boldsymbol{x})^\mathsf{T} = \begin{bmatrix} 1 & x \end{bmatrix}$ *are linearly independent and all the subsequent monomials are linearly dependent on them. In the terminology border basis techniques, the set of dependent monomials* $\{y, x^2, xy\}$ *is called the border, as visualized in Figure 1 [17].*

Notice that, since the border comprises all first-degree multiples of $\boldsymbol{b}(\boldsymbol{x})$*, any other monomial, and by extension any polynomial, can be reduced to a linear combination of the monomials* $\boldsymbol{b}(\boldsymbol{x})$ *by recursively applying the relations in* (1)*. For example:* $y^2 = y(y) = y(-1-x) = -y - xy = -(-1-x) - 2 = -1 + x$*. In a sense, the other monomials can be represented in terms of the basis elements* $\boldsymbol{b}(\boldsymbol{x})$ *in the common roots.*

Multiplication by x *and* y *modulo these relations is then a linear map from the elements of* $\boldsymbol{b}(\boldsymbol{x})$ *onto itself and can thus be captured by so-called multiplication*

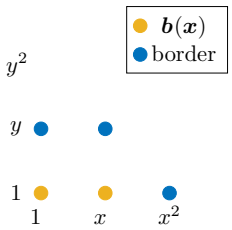

Figure 1: Visual representation of the independent monomials and the dependent monomials in Example 2.1.

matrices $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathbb{R}^{2\times 2}$*:*

$$\begin{bmatrix} x \\ x^2 \end{bmatrix} = \begin{matrix} 1 & x \\ \end{matrix} \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = \boldsymbol{A}_1 \begin{bmatrix} 1 \\ x \end{bmatrix},$$
$$\begin{bmatrix} y \\ xy \end{bmatrix} = \begin{matrix} 1 & x \\ \end{matrix} \begin{bmatrix} -1 & -1 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = \boldsymbol{A}_2 \begin{bmatrix} 1 \\ x \end{bmatrix}.$$

The rows of these matrices correspond to the coefficients w.r.t. $\boldsymbol{b}(\boldsymbol{x})$ *of* $x \cdot \boldsymbol{b}(\boldsymbol{x})$ *and* $y \cdot \boldsymbol{b}(\boldsymbol{x})$*, respectively. Indeed, these matrices express multiplication on the linear span of* $\boldsymbol{b}(\boldsymbol{x})$ *by acting on the coefficient vectors:*

$$x(1+2x) = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ x^2 \end{bmatrix} = \begin{matrix} 1 & x \\ \end{matrix} \begin{bmatrix} 1 & 2 \end{bmatrix} \boldsymbol{A}_1 \begin{bmatrix} 1 \\ x \end{bmatrix} = \begin{matrix} 1 & x \\ \end{matrix} \begin{bmatrix} -4 & 7 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}.$$

Lastly, since $xy = yx$*, it necessarily holds that* $\boldsymbol{A}_1\boldsymbol{A}_2 = \boldsymbol{A}_2\boldsymbol{A}_1$*, i.e. the multiplication matrices commute. Moreover, using a joint Schur decomposition [4] the eigenvalues of* $\boldsymbol{A}_1$ *and* $\boldsymbol{A}_2$ *can be paired to obtain the common roots of* $f(x, y)$ *and* $g(x, y)$*:* $(1, -2)$ *and* $(2, -3)$*.*

Example 2.1 illustrates the general principle of eigenvalue-based root finding methods: given a set of $m-$variate polynomials $f_1(\boldsymbol{x}), \ldots, f_s(\boldsymbol{x})$, with $\boldsymbol{x} = (x_1, \ldots, x_m)$, new polynomials in the ideal $I = \{\sum_{i=1}^{s} h_i f_i \mid h_i(\boldsymbol{x}) \in \mathbb{C}[\boldsymbol{x}]\}$ are first derived by considering polynomial multiples of the current set of equations, and linear combinations thereof until a suitable representation of I is obtained. From this representation, a basis for the coordinate ring $\mathbb{C}[\boldsymbol{x}]/I$ can then be constructed: in Example 2.1 $\boldsymbol{b}(\boldsymbol{x})$ constitutes a set of basis polynomials for $\mathbb{C}[\boldsymbol{x}]/I$. Note that if the common roots of $f_1, \ldots, f_s$, counted with multiplicity, are finite in number, $\mathbb{C}[\boldsymbol{x}]/I$ forms a vector space of dimension equal to this number [5, 17].

relevant? To appendix?

Once such a basis $\boldsymbol{b}(\boldsymbol{x})$ is obtained, the multiplication maps can be constructed by considering the coefficients of the elements of $x_i \cdot \boldsymbol{b}(\boldsymbol{x})$ w.r.t. $\boldsymbol{b}(\boldsymbol{x})$. The eigenvalues of these multiplication maps coincide with specific coordinates of the

common roots: Let $\boldsymbol{b}(\boldsymbol{x}) \in \mathbb{C}^q[\boldsymbol{x}]$ contain the basis polynomials in any order. By construction, it then holds that $x_i \boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{A}_i \boldsymbol{b}(\boldsymbol{x})$ for any common root $\boldsymbol{x}$, such that x_i is clearly an eigenvalue of $\boldsymbol{A}_i$. Conversely, for any eigenvalue λ_j of $\boldsymbol{A}_i$ and a corresponding left eigenvector $\boldsymbol{v}_j^\mathsf{T}$, it holds that $x_i \boldsymbol{v}_j^\mathsf{T} \boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{v}_j^\mathsf{T} \boldsymbol{A}_i \boldsymbol{b}(\boldsymbol{x}) = \lambda_j \boldsymbol{v}_j^\mathsf{T} \boldsymbol{b}(\boldsymbol{x})$ so that $(x_i - \lambda_j) \boldsymbol{v}_j^\mathsf{T} \boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{0}$. Since $\boldsymbol{v}_j^\mathsf{T} \boldsymbol{b}(\boldsymbol{x})$ is a nonzero polynomial in $\mathbb{C}[\boldsymbol{x}]/I$, it does not lie in I and thus does not vanish in all common roots $\boldsymbol{x}$ at once. Hence, there must exist at least one common root $\boldsymbol{x}$ for which $x_i = \lambda_j$ [17]. Pairwise commuting matrices have a common left eigenvector associated with each eigenvalue, so that the eigenvalues corresponding to $\boldsymbol{v}_j^\mathsf{T}$ of the other multiplication maps correspond to the other coordinates of the common root.

The way new equations are obtained distinguishes different algebraic root finding techniques: The current equations may be 'shifted' and subsequently 'reduced' onto $\boldsymbol{b}(\boldsymbol{x})$ using e.g. a reduced row echelon form of the corresponding coefficient matrix to obtain a new set of generators for I. Such approaches are prevalent in Gröbner and border basis techniques [17, 10, 14, 21, 12, 16].

In resultant-based methods, the considered equations are simply shifted using predefined shifts or until a 'gap' is detected, without ever updating the set of generators for I [1, 7, 19, 20, 9, 8]. A drawback of the latter approach is that it often leads to large matrices in the sub steps. However, these methods often do not construct the basis for $\mathbb{C}[\boldsymbol{x}]/I$ explicitly. This allows for flexibility in the choice basis and allows one to delegate this choice to standard numerical linear algebra techniques, which often benefits numerical stability [19]. Indeed, to construct the ith multiplication map, it suffices to know a full column-rank coefficient matrix of a set of polynomials $\boldsymbol{p}(\boldsymbol{x})$ w.r.t. this unknown basis, and the coefficients of $x_i \boldsymbol{p}(\boldsymbol{x})$ w.r.t. the same basis. This allows one to formulate an overdetermined linear system to find $\boldsymbol{A}_i$, which can be tackled with standard numerical linear algebra techniques without any bookkeeping of the underlying basis polynomials. [7]. Example 2.2 briefly illustrates this approach.

Example 2.2 *Reconsider the equations from Example 2.1. It was already derived that* $y^2 = -1 + x$ *and* $xy = 2 - 4x$ *holds in the common roots, such that:*

$$\begin{array}{r} \\ x \\ x^2 \\ xy \end{array}\!\!\begin{bmatrix} 1 & x \\ 0 & 1 \\ -2 & 3 \\ 2 & -4 \end{bmatrix}\begin{bmatrix} 1 \\ x \end{bmatrix} = \begin{array}{r} 1 \\ x \\ y \end{array}\!\!\begin{bmatrix} 1 & x \\ 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix} \boldsymbol{A}_1 \begin{bmatrix} 1 \\ x \end{bmatrix},$$

$$\begin{array}{r} \\ y \\ xy \\ y^2 \end{array}\!\!\begin{bmatrix} 1 & x \\ -1 & -1 \\ 2 & -4 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ x \end{bmatrix} = \begin{array}{r} 1 \\ x \\ y \end{array}\!\!\begin{bmatrix} 1 & x \\ 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix} \boldsymbol{A}_2 \begin{bmatrix} 1 \\ x \end{bmatrix}.$$

Evaluating the equation above in the common roots $(1, -2)$ *and* $(2, -3)$ *results in a matrix equation where* $\boldsymbol{B} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{matrix} 1 \\ x \end{matrix}$ *contains the evaluated basis elements*

$\{1, x\}$ *as its columns. Since we can see that they result in an invertible matrix. Right multiplying with* $\boldsymbol{B}^{-1}$ *then eliminates* $\boldsymbol{B}$ *from these equations, leading to a set of linear equations that can be used to determine* $\boldsymbol{A}_1$ *and* $\boldsymbol{A}_2$*, and subsequently, the common roots. Since only* $\boldsymbol{B}$ *depends on the roots and can always eliminated from the equations, the solutions do not need to be known beforehand. When there are multiplicities involved, the argument is slightly more involved, but the core idea holds for any set of polynomials with a zero-dimensional solutions set [17].*

Moreover, a change of basis for $\mathbb{C}[\boldsymbol{x}]/I$ *results in a similarity transform of the multiplication maps. Indeed, it was already illustrated in Example 2.1 that the rows of the multiplication maps correspond to the coefficients of* $x \cdot \boldsymbol{b}(\boldsymbol{x})$ *and* $y \cdot \boldsymbol{b}(\boldsymbol{x})$ *w.r.t. the basis* $\boldsymbol{b}(\boldsymbol{x})$*, so that changing this basis may affect the entries. They can, however, still be determined in similar fashion given that the necessary coefficients w.r.t. the new basis are known. For example, let* $\boldsymbol{P} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ *and consider the basis* $\begin{bmatrix} 1+x & 1-x \end{bmatrix} = \begin{bmatrix} 1 & x \end{bmatrix} \boldsymbol{P}^\mathsf{T}$ *and construct* $\boldsymbol{A}_1$*:*

$$\begin{array}{r} \\ x \\ x^2 \\ xy \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & x \end{array} \\ \begin{bmatrix} 0 & 1 \\ -2 & 3 \\ 2 & -4 \end{bmatrix} \end{array}\!\!\boldsymbol{P}^{-1}\begin{bmatrix} 1+x \\ 1-x \end{bmatrix} = \begin{array}{r} \\ 1 \\ x \\ y \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & x \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix} \end{array}\!\!\boldsymbol{P}^{-1}\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^{-1}\begin{bmatrix} 1+x \\ 1-x \end{bmatrix}.$$

$$\leftrightarrow \begin{array}{r} \\ x \\ x^2 \\ xy \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1+x & 1-x \end{array} \\ \begin{bmatrix} 0.5 & -0.5 \\ 0.5 & -2.5 \\ -1 & 3 \end{bmatrix} \end{array}\!\!\begin{bmatrix} 1+x \\ 1-x \end{bmatrix} = \begin{array}{r} \\ 1 \\ x \\ y \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1+x & 1-x \end{array} \\ \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \\ -1 & 0 \end{bmatrix} \end{array}\!\!\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^{-1}\begin{bmatrix} 1+x \\ 1-x \end{bmatrix}.$$

Clearly, $\boldsymbol{P}\boldsymbol{A}_1\boldsymbol{P}^{-1}$ *acts like a multiplication map in this new basis. While the multiplication maps depend on the specific basis used, their eigenvalues do not, such that we can essentially abstract the column labels to some underlying basis in the equations above, instead of considering these polynomials explicitly. This illustrates that it suffices to have a vector representation of a satisfactory set of polynomials and their image under multiplication by* x *to construct* $\boldsymbol{A}_1$ *w.r.t. the same basis. A similar argument holds for* $\boldsymbol{A}_2$*.*

Moreover, since left multiplication with an invertible matrix does not affect the solutions of this linear equation, the specific row labels do also not necessarily matter to obtain the multiplication maps.

In this work, we aim to further bridge the gap between the different algebraic techniques by introducing a purely numerical linear algebra-based, basis-agnostic approach to compute the multiplication maps that compresses the number of polynomials under consideration. Moreover, the presented approach further simplifies the required 'bookkeeping' in the sense that the underlying polynomials that are represented by each vector do not need to be known explicitly, resulting a fairly straightforward algorithm.

3 Basis-agnostic approach

We now outline the computational steps in the main iteration of the proposed approach. While the derivations concern polynomials, all the computational steps only act upon the coefficient matrices and do not depend on the underlying polynomials. As such, none of these polynomials need to be constructed explicitly, allowing flexibility in the choice of bases. Furthermore, none of the steps rely on explicit gradings of the polynomials or tracking a suitable set of monomials, as is the case for conventional border basis algorithms [13], instead relying on a rank decision to decide upon which new basis elements to add. This reduces the symbolic bookkeeping compared to such algorithms. Remark that, while the underlying polynomials are considered explicitly in this section, they can be abstracted since all the computational steps only rely on the coefficient matrices.

Consider a set of basis polynomials $\boldsymbol{b}(\boldsymbol{x})$ and some equations $\boldsymbol{p}_1(\boldsymbol{x}) = \boldsymbol{N}_0\boldsymbol{b}(\boldsymbol{x})$. As before, these relations can be interpreted as $\boldsymbol{p}_1(\boldsymbol{x})$ being represented in terms of the basis elements $\boldsymbol{b}(\boldsymbol{x})$. Assume a similar representation for their images $x_i\boldsymbol{p}_1(\boldsymbol{x}) = \boldsymbol{N}_i\boldsymbol{b}(\boldsymbol{x})$ for all i is given. Since expressions for $x_i\boldsymbol{p}_1(\boldsymbol{x})$ w.r.t. the current basis $\boldsymbol{b}(\boldsymbol{x})$ are known, row $\boldsymbol{N}_0$ can be thought of as the $\boldsymbol{b}(\boldsymbol{x})$-coefficient space of polynomials whose images under multiplication with x_i can be represented in terms of the current 'working basis' $\boldsymbol{b}(\boldsymbol{x})$ again. Its orthogonal complement is then the coefficient space of polynomials with multiples that cannot be expressed in terms of $\boldsymbol{b}(\boldsymbol{x})$.

We now aim to derive a 'richer' set of equations that includes expressions for all first-degree multiples of $\boldsymbol{b}(\boldsymbol{x})$. By the given relations, the multiples of a polynomial of $\boldsymbol{v}^\mathsf{T}\boldsymbol{b}(\boldsymbol{x}) = (\boldsymbol{v}_1 + \boldsymbol{v}_2)^\mathsf{T}\boldsymbol{b}(\boldsymbol{x})$, where $\boldsymbol{v}_1 \in \text{row } \boldsymbol{N}_0$ and $\boldsymbol{v}_2 \in \text{null } \boldsymbol{N}_0$, are of the form $x_i\boldsymbol{v}^\mathsf{T}\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{v}_3\boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{v}_2x_i\boldsymbol{b}(\boldsymbol{x})$, with $\boldsymbol{v}_3 \in \text{row } \boldsymbol{N}_i$. Since the current equations do not provide an expression for $\boldsymbol{v}_2x_i\boldsymbol{b}(\boldsymbol{x})$ in terms of $\boldsymbol{b}(\boldsymbol{x})$, we extend the working basis to $\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})^\mathsf{T} & x_i\boldsymbol{p}_2(\boldsymbol{x})^\mathsf{T}\end{bmatrix}$, where $\boldsymbol{p}_2(\boldsymbol{x}) = \boldsymbol{Q}\boldsymbol{b}(\boldsymbol{x})$ and $\boldsymbol{Q}^\mathsf{T}$ is a basis for null $\boldsymbol{N}_0$. Note that the number of basis elements to add thus follows from rank $\boldsymbol{N}_0$.

notation: transpose before brackets?

One can then define the maps $\boldsymbol{A}_i$ as follows:

$$\begin{aligned} x_i\begin{bmatrix}\boldsymbol{N}_0\\ \boldsymbol{Q}\end{bmatrix}\boldsymbol{b}(\boldsymbol{x}) = x_i\begin{bmatrix}\boldsymbol{p}_1(\boldsymbol{x})\\ \boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} &= \begin{bmatrix}\boldsymbol{N}_i & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} \\ \rightarrow x_i\boldsymbol{b}(\boldsymbol{x}) = \begin{bmatrix}\boldsymbol{N}_0{}^\dagger & \boldsymbol{Q}^\dagger\end{bmatrix}\begin{bmatrix}\boldsymbol{N}_i & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} &= \boldsymbol{A}_i\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}. \end{aligned} \tag{2}$$

The maps $\boldsymbol{A}_i$ express multiplication w.r.t. $\boldsymbol{b}(\boldsymbol{x})$-coefficients in the following sense: $x_i\boldsymbol{v}^\mathsf{T}\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{v}^\mathsf{T}x_i\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{v}^\mathsf{T}\boldsymbol{A}_i\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = \begin{bmatrix}(\boldsymbol{A}_i)_1 & (\boldsymbol{A}_i)_2\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}$. Also remark that $\boldsymbol{A}_i$ may map the coefficients $\boldsymbol{v}^\mathsf{T}$ to a higher dimensional vector space, to account for the new basis polynomials that were added to express the unknown images of $\boldsymbol{p}_2(\boldsymbol{x})$.

Now, the multiplication property of the matrices $\boldsymbol{A}_i$ can be used to construct new linear relations between the basis elements $\boldsymbol{b}(\boldsymbol{x})$ and $x_i\boldsymbol{p}_2(\boldsymbol{x})$ as follows:

$$\begin{aligned}\boldsymbol{0} =& x_j\left(x_i\boldsymbol{b}(\boldsymbol{x}) - \boldsymbol{A}_i\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}\right) = \boldsymbol{A}_j\begin{bmatrix}x_i\boldsymbol{b}(\boldsymbol{x})\\ x_ix_j\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} - \boldsymbol{A}_i\begin{bmatrix}x_j\boldsymbol{b}(\boldsymbol{x})\\ x_ix_j\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}\\ =&(\boldsymbol{A}_j)_1x_i\boldsymbol{b}(\boldsymbol{x}) + (\boldsymbol{A}_j)_2x_jx_i\boldsymbol{p}_2(\boldsymbol{x}) - (\boldsymbol{A}_i)_1x_j\boldsymbol{b}(\boldsymbol{x}) - (\boldsymbol{A}_i)_2x_jx_i\boldsymbol{p}_2(\boldsymbol{x})\\ =&((\boldsymbol{A}_j)_1\begin{bmatrix}(\boldsymbol{A}_i)_1 & (\boldsymbol{A}_i)_2 & \boldsymbol{0}\end{bmatrix} - (\boldsymbol{A}_i)_1\begin{bmatrix}(\boldsymbol{A}_j)_1 & \boldsymbol{0} & (\boldsymbol{A}_j)_2\end{bmatrix})\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\\ x_j\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}\\ =&\begin{bmatrix}(\boldsymbol{C}_{i,j})_0 & (\boldsymbol{C}_{i,j})_1 & (\boldsymbol{C}_{i,j})_2\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\\ x_j\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = \boldsymbol{C}_{i,j}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\\ x_j\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix},\end{aligned} \tag{3}$$

where the fourth equality holds as $(\boldsymbol{A}_i)_2 = (\boldsymbol{A}_j)_2, \forall i,j \in \{1,\dots,m\}$ by their construction in (2). The elements of $\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})^\mathsf{T} & x_i\boldsymbol{p}_2(\boldsymbol{x})^\mathsf{T} & x_j\boldsymbol{p}_2(\boldsymbol{x})^\mathsf{T}\end{bmatrix}$ thus satisfy any of the linear relations in row $\boldsymbol{C}_{i,j}$ and they can thus be written as a linear combination of a smaller number of polynomials whenever $\boldsymbol{C}_{i,j}$ is nonzero. To construct such a new basis, consider $\boldsymbol{C}$ defined below, which captures the relations over generated by all (i,j)-pairs. To this end, define the set of pairs $\mathcal{A} = \{(i,j) \in \{1\dots,m\}^2 \mid j > i\}$, then construct the block-matrix $\boldsymbol{C}$ using row multi-indices $\boldsymbol{\alpha} \in \mathcal{A}$:

$$\begin{aligned}\forall\boldsymbol{\alpha} \in \mathcal{A} : \boldsymbol{C}_{\boldsymbol{\alpha},1} = (\boldsymbol{C}_{\boldsymbol{\alpha}})_0;\ \boldsymbol{C}_{\boldsymbol{\alpha},\alpha_1} = (\boldsymbol{C}_{\boldsymbol{\alpha}})_1;\ \boldsymbol{C}_{\boldsymbol{\alpha},\alpha_2} = (\boldsymbol{C}_{\boldsymbol{\alpha}})_2,;\ \boldsymbol{C}_{\boldsymbol{\alpha},\{1,\dots,m\}\setminus\alpha} = \boldsymbol{0},\\ \boldsymbol{T} = \text{basis}(\text{null }\boldsymbol{C}).\end{aligned} \tag{4}$$

Equation (3) then implies that the current basis polynomials can be expressed as a set of polynomials in range $\boldsymbol{T}$. Indeed, let the rows of $\boldsymbol{R}$ form a basis for row $\boldsymbol{C}$ and define the following polynomials:

$$\begin{aligned}&\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{T}^\dagger\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_1\boldsymbol{p}_2(\boldsymbol{x})\\ \vdots\\ x_m\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} \overset{(3)}{=} \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{b}_{\text{new}}(\boldsymbol{x})\end{bmatrix}\\ \leftrightarrow \begin{bmatrix}\boldsymbol{R}^\dagger & \boldsymbol{T}\end{bmatrix}\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{b}_{\text{new}}(\boldsymbol{x})\end{bmatrix} &= \begin{bmatrix}\boldsymbol{T}_0\\ \boldsymbol{T}_1\\ \vdots\\ \boldsymbol{T}_m\end{bmatrix}\boldsymbol{b}_{\text{new}}(\boldsymbol{x}) = \begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_1\boldsymbol{p}_2(\boldsymbol{x})\\ \vdots\\ x_m\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}.\end{aligned} \tag{5}$$

More generally, the left inverse of $\boldsymbol{T}$ is only defined up the left kernel of $\boldsymbol{T}$ so that in theory, the basis $\boldsymbol{b}_{\text{new}}(\boldsymbol{x})$ can be chosen modulo the relations in row $\boldsymbol{C}$.

Finally, by (2), it then holds that:

$$x_i\boldsymbol{b}(\boldsymbol{x}) = \boldsymbol{A}_i\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})\\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = \boldsymbol{A}_i\begin{bmatrix}\boldsymbol{T}_0\\ \boldsymbol{T}_i\end{bmatrix}\boldsymbol{b}_{\text{new}}(\boldsymbol{x}) = \boldsymbol{N}_i\boldsymbol{b}_{\text{new}}(\boldsymbol{x}), \tag{6}$$

leaving us with matrix representations for both $x_i\boldsymbol{b}(\boldsymbol{x})$, $\boldsymbol{b}(\boldsymbol{x})$ and all $x_i\boldsymbol{p}_2(\boldsymbol{x})$ w.r.t. $\boldsymbol{b}_{\text{new}}(\boldsymbol{x})$. This allows us to construct maps again through (2) by choosing, $\boldsymbol{p}_1(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})$, $\boldsymbol{N}_0 = \boldsymbol{T}_0$ and $\boldsymbol{N}_i$ as defined in (6), so that the whole procedure can be repeated. The computational steps are summarized in Algorithm 1 in Section 4, where the convergence condition of the algorithm is discussed.

Notation (transpose for null basis) consistent?

4 Initialization and convergence

In this section, behavior of the procedure presented in Section 3 is further investigated under the assumption that the initial set of polynomials has a finite number of projective solutions. First, an adequate initialization of the algorithm is illustrated, which is of major importance to ensure convergence. We emphasize that a convergence proof remains an active research question. Nonetheless, valuable insights are provided for the case where monomial bases are chosen. Intuitively, the general case can then be achieved through a simple change of basis.

Notice that whenever $\boldsymbol{b}(\boldsymbol{x}), \boldsymbol{p}_2(\boldsymbol{x})$ consist of monomials, by the second part of (5), $\boldsymbol{b}_{\text{new}}(\boldsymbol{x})$ does as well if $\boldsymbol{T}^{\mathsf{T}}$ is chosen to be in reduced row echelon form: The identity rows of $\boldsymbol{T}$ select some set of monomials out of those in $\boldsymbol{b}(\boldsymbol{x}), x_1\boldsymbol{p}_2(\boldsymbol{x}), \ldots, x_m\boldsymbol{p}_2(\boldsymbol{x})$. The use of this particular null space basis to obtain a monomial basis is briefly sketched in Example 4.1, which illustrates the initialization of the algorithm as well.

Example 4.1 *Consider the following set of bivariate polynomial equations:*

$$\begin{aligned} f(x,y) &= 1 + 3x + y + 2y^2 = 0, \\ g(x,y) &= 1 + 2x + y + 3x^2 + xy + 3y^2 + xy^2 + 3y^3 = 0. \end{aligned}$$

To enforce the relations implied by $f(x,y)$ on the third-degree monomials involved, the shifts $xf(x,y)$ and $yf(x,y)$ are considered as well. In border basis terms, this allows the polynomials to be 'reduced' among themselves first, avoiding ambiguity on how to reduce the linearly dependent monomials, ensuring that their reductions are well-defined [13]. These equations can be summarized in the following coefficient matrix, a so-called Macaulay matrix [1]. Remark that from here on, we adopt a notation that better highlights the role of the coefficient matrices and the implicit nature of the underlying polynomial objects, which are only indicated using row and column labels, as in Examples 2.1 and 2.2:

$$\boldsymbol{M} = \begin{array}{r} \\ f(x,y) = 0 \\ xf(x,y) = 0 \\ yf(x,y) = 0 \\ g(x,y) = 0 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 1 & x & y & x^2 & xy & y^2 & x^3 & x^2y & xy^2 & y^3 \end{array} \\ \begin{bmatrix} 1 & 3 & 1 & & & 2 & & & & \\ & 1 & & 3 & 1 & & & & 2 & \\ & & 1 & & 3 & 1 & & & & 2 \\ 1 & 2 & 1 & 3 & 1 & 3 & & & 2 & 1 \end{bmatrix} \end{array}. \tag{7}$$

Taking a basis $\boldsymbol{N}$ for the null space of $\boldsymbol{M}$ is a straightforward way to obtain the linear relations between the different basis monomials that are implied by the

equations described in (7)*. By a similar argument as in* (5)*, the dimension of this null space indicates that the number of underlying basis elements needed to describe all the current basis elements, i.e. monomials up to degree three. Here,* $\boldsymbol{N}^\mathsf{T}$ *is chosen in reduced row-echelon form to guarantee that the underlying basis* $\boldsymbol{b}(\boldsymbol{x})$*, i.e., the set of column labels of* $\boldsymbol{N}$*, consists of monomials:*

$$\boldsymbol{N} = \begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \\ x^3 \\ x^2y \\ xy^2 \\ y^3 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^2 & x^3 & x^2y \end{array} \\ \left[\begin{array}{cccccc} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ -\frac{1}{6} & -\frac{11}{6} & -\frac{1}{2} & & & \\ -\frac{1}{2} & -\frac{3}{2} & -\frac{1}{2} & & & \\ & & & & 1 & \\ & & & & & 1 \\ \frac{1}{12} & \frac{5}{12} & \frac{1}{4} & -\frac{3}{2} & & \\ \frac{1}{2} & \frac{7}{2} & \frac{1}{2} & & & \end{array}\right] \end{array}.$$

In general, the choice of null space basis only affects the underlying polynomial (implicit) basis $\boldsymbol{b}(\boldsymbol{x})$*, as already pointed out in Example 2.2.*

Among these monomials we can now select $\boldsymbol{p}_1(\boldsymbol{x})^\mathsf{T} = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix}$*. Notice that the images of the third-degree monomials are not immediately known. However, since* xy^2 *and* y^3 *can be expressed in terms of* $\boldsymbol{p}_1(\boldsymbol{x})$*, their images under multiplication are linear combinations of those of* $\boldsymbol{p}_1(\boldsymbol{x})$*, which are known. This leaves us with* x^3 *and* x^2y*, as can be easily revealed using any basis* $\boldsymbol{Q}$ *for the null space of*[1] $\boldsymbol{N}_{\mathcal{I},:}$*, with* $\mathcal{I}_0 = \{1, x, y, x^2, xy, y^2\}$*, that is, the rows corresponding to the elements of* $\boldsymbol{p}_1(\boldsymbol{x})$*, e.g.:*

$$\boldsymbol{Q} = \begin{array}{c} \\ x^3 \\ y^3 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^2 & x^3 & x^2y \end{array} \\ \begin{bmatrix} & & & & 1 & \\ & & & & & 1 \end{bmatrix} \end{array}.$$

Note the similarity with choosing taking the kernel of $\boldsymbol{T}_0$ *to choose* $\boldsymbol{p}_2(\boldsymbol{x})$ *in* (2)*. Consequently, we choose* $\boldsymbol{p}_2(\boldsymbol{x})^\mathsf{T} = \boldsymbol{b}(\boldsymbol{x})^\mathsf{T}\boldsymbol{Q}^\mathsf{T}\begin{bmatrix} x^3 & x^2y \end{bmatrix}$ *and the x-images* $\{x^4, x^3y\}$ *and y-images* $\{x^3y, x^2y^2\}$ *of these independent basis elements are introduced to define maps as in* (2)*, where we choose* $\boldsymbol{N}_0^\mathsf{T} = \boldsymbol{N}_{\mathcal{I}_0,:}$ *and* $\boldsymbol{N}_1 = \boldsymbol{N}_{\mathcal{I}_1,:}$*,* $\boldsymbol{N}_2 = \boldsymbol{N}_{\mathcal{I}_2,:}$*, where*[2] $\mathcal{I}_1 = x \cdot \mathcal{I}_0$ *and* $\mathcal{I}_2 = y \cdot \mathcal{I}_0$*:*

[1] *Notation:* ':' indicates 'all indices' and for convenience, we index the rows of $\boldsymbol{N}$ using set of the corresponding monomials, as the precise ordering of the rows does not matter in general.

[2] *Notation:* $x \cdot \mathcal{I} = \{ix \mid i \in \mathcal{I}\}$.

$$\boldsymbol{A}_1 = \begin{array}{c} \\ x \\ x^2 \\ xy \\ x^3 \\ x^4 \\ xy^3 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & x & y & x^2 & x^3 & y^3 & x^4 & x^3y \end{array} \\ \left[\begin{array}{cccccccc} -0.0097 & 0.9129 & -0.0290 & 0.2130 & & 0.1162 & & \\ 0.0238 & 0.2139 & 0.0713 & 0.4771 & & -0.2852 & & \\ -0.1631 & -1.8016 & -0.4894 & -0.0775 & & -0.0423 & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right] \end{array}$$

$$\boldsymbol{A}_2 = \begin{array}{c} \\ y \\ xy \\ y^2 \\ x^2y \\ x^3y \\ y^4 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & x & y & x^2 & x^3 & y^3 & x^3y & x^2y^2 \end{array} \\ \left[\begin{array}{cccccccc} -0.0549 & -0.4293 & 0.9129 & -0.1743 & & & & \\ -0.0320 & -0.7796 & -0.2861 & 0.4278 & & & & \\ -0.4800 & -1.3439 & -0.4683 & 0.0634 & & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right] \end{array}$$

Notice that the rows of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ different vector representations for $\{x, y, x^2, xy, y^2\}$ compared to the ones provided by $\boldsymbol{N}$. This discrepancy will now be addressed next for the general case. Moreover, contrary to border basis methods, lower degree monomials are mapped onto higher degree ones as well.

To reduce the number of equations under consideration, in each iteration, the equations $\boldsymbol{p}_1(\boldsymbol{x}) = \boldsymbol{N}_0\boldsymbol{b}(\boldsymbol{x})$ and $x_i\boldsymbol{p}_1(\boldsymbol{x}) = \boldsymbol{N}_i\boldsymbol{b}(\boldsymbol{x})$, are 'compressed' into the shift equations (2), which capture how the basis elements are mapped under multiplication, modulo the ideal. Similar to Macaulay-matrix methods [1, 7], this compression can be achieved through solving a (inconsistent) linear system for $\boldsymbol{A}_i$. The equation seems to entail a loss of information, as the right-hand-side equations do not imply the left-hand-side ones. However, due to known relations between the elements of $\boldsymbol{p}_1(\boldsymbol{x})$, the representation of $x_i\boldsymbol{p}_1(\boldsymbol{x})$ is not unique: Let the rows of $\boldsymbol{L}$ form an orthonormal basis for the null space of $\begin{bmatrix}\boldsymbol{N}_0^\mathsf{T} & \boldsymbol{Q}^\mathsf{T}\end{bmatrix}$. Then from the left-hand-side equation of (2), it also follows that:

setting the discrepancy to zero = the info??

$$\boldsymbol{0} = x_i\boldsymbol{L}\boldsymbol{N}_0\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = \boldsymbol{L}\begin{bmatrix}\boldsymbol{N}_i & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = \boldsymbol{Z}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}.$$

However, $\boldsymbol{Z}$ is generally nonzero, so that this describes nontrivial relations between the basis elements $\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x})^\mathsf{T} & x_i\boldsymbol{p}_2(\boldsymbol{x})^\mathsf{T}\end{bmatrix}$. By the nature of the pseudoinverse we have that null $\boldsymbol{N}_0{}^\dagger$ = row $\boldsymbol{L}$, such that multiplication maps $\boldsymbol{A}_i$ essentially describe the left-hand side equation of (2), modulo the relations spanned by $\boldsymbol{Z}$:

$$x_i\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{N}_0\boldsymbol{A}_i\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = (\boldsymbol{N}_0\boldsymbol{A}_i + \boldsymbol{L}^\mathsf{T}\boldsymbol{Z})\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix} = \begin{bmatrix}\boldsymbol{N}_i & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x})\end{bmatrix}.$$

Notice the similarity with border basis methods, where descriptions of the images of the basis monomials in terms of the basis elements are used as generating

equations at each iteration [12]. This allows one to rewrite all monomials that are multiples of the border in terms of the basis elements through repeated reduction, as briefly illustrated in Example 1. By analogy, this suggests that, given an appropriate selection of $\boldsymbol{b}(\boldsymbol{x})$, the 'shift equations' $\mathbf{0} = x_i\boldsymbol{b}(\boldsymbol{x}) - \boldsymbol{A}_i \begin{bmatrix} \boldsymbol{b}(\boldsymbol{x}) \\ x_i\boldsymbol{p}_2(\boldsymbol{x}) \end{bmatrix}$ generated in (2) capture all essential relations at each degree. Finally, note that even when a monomial basis and the same ordering is used, these equations may differ from their equivalents in border basis methods. Indeed, as just shown, these shift equations are only defined up to the relations described by row $\boldsymbol{Z}$. We now continue with the Example 4.2 to demonstrate how convergence may be detected after obtaining new equations.

Example 4.2 *From the compressed equations, new relations can then be derived. Let us continue with Example 4.1 and derive new equations using* (3)*, where* $(\boldsymbol{A}_i)_1 = (\boldsymbol{A}_i)_{:,\mathcal{I}_0}$*, by constructing* $\boldsymbol{C}$ *and a basis for its null space* $\boldsymbol{T}$*, again in reduced 'column echelon form' to guarantee a monomial basis:*

$$\boldsymbol{C}^\intercal = \begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ x^3 \\ y^3 \\ x^4 \\ x^3y \\ x^3y \\ x^2y^2 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \end{array} \\ \begin{bmatrix} 0.1439 & -0.0708 & 0.2524 & & & \\ 1.1181 & -0.5923 & 2.0144 & & & \\ 0.2194 & -0.1583 & 0.4484 & & & \\ 0.6776 & 0.4485 & -0.0662 & & & \\ 0.1743 & -0.4278 & -0.0634 & & & \\ 0.1355 & 0.2464 & -0.4248 & & & \\ & & & & & \\ & & & -1 & & \\ & & & 1 & & \\ 0.1162 & -0.2852 & -0.0423 & & & \end{bmatrix} \end{array},$$

$$\boldsymbol{T} = \begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ x^3 \\ y^3 \\ x^4 \\ x^3y \\ x^3y \\ x^2y^2 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^3 & x^4 & x^3y \end{array} \\ \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ -0.3148 & -2.4630 & -0.5000 & & & \\ & & & 1 & & \\ 0.6605 & 5.2654 & 1.1667 & & & \\ & & & & 1 & \\ & & & & & 1 \\ & & & & & 1 \\ -0.1728 & -1.4012 & -0.3333 & -1.5 & & \end{bmatrix} \end{array}.$$

First, notice that the fourth row of $\boldsymbol{C}$ *eliminates the duplicate 'dimension' that was added for* x^3y*. While it was obvious in this case that the underlying basis was of too high dimension, for more general bases, the images of different basis elements are not necessarily equal so that four new dimensions may indeed be required*[3]*. Furthermore, the row labels are not known in practice, so that this worst-case scenario should be considered.*

Next, the dimension of range $\boldsymbol{T}$ *is six, indicating that there are again six under-*

[3]Consider the case where $\boldsymbol{p}_2(\boldsymbol{x})$ comprises x^3 and y^3.

lying basis elements by (5)*, just like during the initialization. This stagnation of the number of basis elements indicates 'regularity' [15]. Indeed, by Bezout's theorem, we expect there to be exactly six solutions and as explained in Section 2, the coordinate ring is expected to of the same dimension. Remark that, had* $\{xy^2, y^3\}$ *be considered part of* $\boldsymbol{p}_2(\boldsymbol{x})$ *in Example 4.1, eight new dimensions would have to be added, leading to a nullity of at least eight and thus hampering the convergence of the algorithm. This demonstrates the importance of correctly choosing the set of polynomials* $\boldsymbol{p}_2(\boldsymbol{x})$ *with 'unknown images'.*

Let us now look into the new equations that are generated in each iteration. Reconsider the equations defined by row $\boldsymbol{C}$ in (3). Notice that these multiples of the shift equations:

$$x_j \left(x_i \boldsymbol{b}(\boldsymbol{x}) - \boldsymbol{A}_i \begin{bmatrix} \boldsymbol{b}(\boldsymbol{x}) \\ x_i \boldsymbol{p}_2(\boldsymbol{x}) \end{bmatrix} \right) = \boldsymbol{0},$$

reduced through the shift equations themselves. Again, there is a strong analogy with border basis methods, where only multiples of the shift equations are used to generate new relations at each degree by reducing them [12, 13]. In the current approach, the use of $\boldsymbol{T}$ in (5) then selects new independent basis monomials modulo the obtained equations and rewrites the other in terms of this basis.

There may be additional relations in row $\boldsymbol{C}$, however: consider a basis monomial $b(\boldsymbol{x})$ such that both $x(\boldsymbol{x})$ and $yb(\boldsymbol{x})$ are elements of $\boldsymbol{p}_2(\boldsymbol{x})$. In such a case, too many new basis elements are added: both $yxb(\boldsymbol{x})$ and $xyb(\boldsymbol{x})$ are added in (2). Due to the monomial structure of the basis, it can be worked out that[4] an equation of the form $0 = yxb(\boldsymbol{x}) - xyb(\boldsymbol{x})$ is spanned by the rows of $\boldsymbol{C}$, as illustrated in Example 4.2. As such, any such 'spurious dimension' is removed.

Finally, Example 4.3 illustrates how the projective multiplication maps can be distilled once convergence is detected. Contrary to border basis algorithms, the solutions at infinity have to be taken into account in the presented approach. The full procedure is also summarized in Algorithm 1. Again, notice in this algorithm that the main iteration only involves standard linear algebra operations on the coefficient matrices.

Example 4.3 *Continuing from example 4.2, now that regularity has been detected, the multiplication maps can be constructed. Without going into too much detail, the solutions at infinity can be accounted for by 'homogenizing'*[5] *the equations by introducing a homogenization variable e.g.* z*, for which there also exists a multiplication map* $\boldsymbol{A}_0$ *[1, 6]. To construct this map, the approach in [2] takes a generic linear combination of* $\boldsymbol{N}_0 = \boldsymbol{T}_{\mathcal{J}_0,:}$*,* $\boldsymbol{N}_1 = \boldsymbol{A}_1\boldsymbol{T}$ *and* $\boldsymbol{N}_2 = \boldsymbol{A}_2\boldsymbol{T}$*, e.g.*

[4] When $\boldsymbol{T}^\intercal$ is chosen in reduced row-echelon form, $\boldsymbol{Q}$ always consists of identity rows so that its pseudoinverse consists of identity rows that select these new monomials.

[5] The homogenization of a polynomial $f(x, y)$ of degree d is $f(z, x, y) = z^d f(\frac{x}{z}, \frac{y}{z})$. The affine roots (x, y) can then be obtained from the projective roots as $(1, \frac{x}{z}, \frac{y}{z})$, assuming $z \neq 0$.

$\boldsymbol{P} = \boldsymbol{N}_0 + \boldsymbol{N}_1 + 2\boldsymbol{N}_2$, *where* $\mathcal{J}_0 = \{1, x, y, x^2, x^3, y^3\}$. *In general* $\mathcal{J}_0$ *corresponds to the basis elements that were not added during the last iteration, or in this case, the initialization. The maps can subsequently be defined as* $\boldsymbol{A}_i = \boldsymbol{P}^\dagger \boldsymbol{N}_i$, *resulting in:*

$$\boldsymbol{A}_0 = \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^3 & x^4 & x^3y \end{array} \\ \left[\begin{array}{cccccc} 0.5227 & 1.0227 & -0.9773 & & & \\ -0.1591 & -0.2045 & -0.0227 & & & \\ 0.3182 & -0.4091 & 0.5000 & & & \\ -0.1406 & -1.0918 & -0.2500 & & & \\ -0.4733 & -3.6613 & -0.8106 & 1 & & \\ 0.3070 & 2.3765 & 0.5303 & & & \end{array}\right] \end{array}$$

$$\boldsymbol{A}_1 = \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^3 & x^4 & x^3y \end{array} \\ \left[\begin{array}{cccccc} -0.1591 & -0.2045 & -0.0227 & & & \\ 0.0682 & 0.3864 & 0.1136 & & & \\ 0.0455 & 0.4091 & -0.0455 & & & \\ -0.4733 & -3.6613 & -0.8106 & 1 & & \\ 3.5946 & 27.8129 & 6.1742 & -7 & 1 & -2 \\ -1.5606 & -12.0758 & -2.6818 & 3 & & 1 \end{array}\right] \end{array}$$

$$\boldsymbol{A}_2 = \begin{array}{c} \begin{array}{cccccc} 1 & x & y & x^3 & x^4 & x^3y \end{array} \\ \left[\begin{array}{cccccc} 0.3182 & -0.4091 & 0.5000 & & & \\ 0.0455 & 0.4091 & -0.0455 & & & \\ -0.1818 & 0.0000 & 0.2727 & & & \\ 0.3070 & 2.3765 & 0.5303 & & & \\ -1.5606 & -12.0758 & -2.6818 & 3 & & 1 \\ 0.6268 & 4.8496 & 1.0758 & -1.5 & & \end{array}\right] \end{array}$$

It can be verified that these maps commute and that, as expected from multiplication maps [7], these matrices satisfy the the homogenizations $f(z, x, y)$ *and* $g(z, x, y)$*: Evaluating in the matrices results in residuals with a Frobenius norm of* 5.7141e−14 *and* 5.7054e−14*, respectively. Applying the techniques of [4], the six projective solutions of the system can be extracted using a joint Schur decomposition, resulting in the projective common roots* $(0.4913 \pm 0.6609\mathrm{i}, -0.1347 \mp 0.0125\mathrm{i}, 0.3217 \mp 0.3242\mathrm{i})$, $(-0.1645, 0.4513, 0.3566)$, *and* $(0, 1, 0)$ *with multiplicity three. The latter multiple root lies at infinity, as indicated by the value* $z = 0$.

5 Numerical experiments

In this section, we briefly investigate the numerical behavior of a naive MATLAB implementation of Algorithm 1 through experiments and compare it to MacaulayLab [22], a linear-algebra based MATLAB toolbox for multivariate polynomials and multiparameter eigenvalue problems.

The experiments[6] are set up as follows: each run considers a randomly generated

[6]The experiments were run on a System HP EliteBook 845 G8 Notebook PC with an AMD

Algorithm 1 Algorithm to compute projective multiplication maps for a set of polynomial equations with a finite number of projective roots.

function MULTIPLICATION_MAPS($\boldsymbol{f}(\boldsymbol{x})$)
  $m \leftarrow \text{length}(\boldsymbol{x})$
  $d \leftarrow \deg(\boldsymbol{f}(\boldsymbol{x}))$
  $\boldsymbol{M} \leftarrow$ Macaulay matrix of $\boldsymbol{f}$ of degree d
  $\boldsymbol{N} \leftarrow \text{basis null } \boldsymbol{M}$
  $\mathcal{I}_0 \leftarrow \{\boldsymbol{x}^{\boldsymbol{\alpha}} \mid \|\boldsymbol{\alpha}\|_1 \leq d-1\}$
  $\boldsymbol{N}_0 \leftarrow \boldsymbol{N}_{\mathcal{I}_0,:}$
  $\forall i \in \{1,\ldots,m\} : \boldsymbol{N}_i \leftarrow \boldsymbol{N}_{x_i \cdot \mathcal{I}_0,:}$
  $\text{dim} \leftarrow \text{size}(\boldsymbol{N}, 2)$
  while true **do**
    $\boldsymbol{Q}^{\mathsf{T}} \leftarrow \text{basis null } \boldsymbol{N}_0$
    $\forall i \in \{1,\ldots,m\} : \boldsymbol{A}_i \leftarrow \begin{bmatrix} {\boldsymbol{N}_0}^{\dagger} & \boldsymbol{Q}^{\dagger} \end{bmatrix} \begin{array}{c} \begin{matrix} 1 & 2 & \cdots & i+1 & \cdots & m+1 \end{matrix} \\ \begin{bmatrix} \boldsymbol{N}_i & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \cdots & \boldsymbol{0} \end{bmatrix} \end{array}$
    construct $\boldsymbol{T}$ as defined in (4)
    $\boldsymbol{N}_0 \leftarrow \boldsymbol{T}_{1:\text{dim},:}$
    $\forall i \in \{1,\ldots,m\} : \boldsymbol{N}_i \leftarrow \boldsymbol{A}_i \boldsymbol{T}$
    if dim = size($\boldsymbol{T}$,2) **then**
      $\boldsymbol{P} \leftarrow$ random linear combination of $\boldsymbol{N}_i$
      $\forall i \in \{0,\ldots,m\} : \boldsymbol{A}_i \leftarrow \boldsymbol{P}^{\dagger} \boldsymbol{N}_i$
      return $\boldsymbol{A}_i$
    else
      $\text{dim} \leftarrow \text{size}(\boldsymbol{T}, 2)$
    end if
  end while
end function

system $\boldsymbol{p}(\boldsymbol{x})$ of m polynomials of degree d in m variables $\boldsymbol{x}$. Each set of equations is solved using both Algorithm 1, where the solutions are extracted from the multiplication maps using the techniques described in [4] and MacaulayLab. For each set of parameters (m, d), five runs are performed, the average runtime and the geometric mean of the maximal residual $\|\boldsymbol{p}(\boldsymbol{x}_k)\|_2$ across all solutions $\boldsymbol{x}_k$ are subsequently reported in Figures 2 and 3, respectively. Degrees up to $10, 8, 6$, and 4 were considered for $m = 2, 3, 4$ and 5, respectively. Whenever one of the runs failed to find all solutions, the run is counted as failed and no timings and residuals are incorporated in the figures for said value of (m, d). The relative tolerance that was used for rank decisions was 10^7, that is, any singular value lower than $10^7\sigma_1$ is considered to be zero.

add comparison to border-basix?

Looking at runtimes in Figure 2 first, the presented method, when it succeeds, consistently outperforms the MacaulayLab implementation. For $m = 2$, the timings seems to scale slightly better with increasing degree d, as the lines slightly diverge. Figure 3 shows that the MacaulayLab implementation is consistently yields more accurate results for a given problem complexity (m, d). Moreover, the accuracy of Algorithm 1 deteriorates significantly faster with increasing degrees, to the point where it cannot provide the solutions to some of the problems were the MacaulayLab implementation still can[7]. Since a convergence proof has yet to be provided, the possibility of the algorithm not converging cannot be entirely ruled out. However, the gradual loss in accuracy suggests that numerical inaccuracies lie at the heart of the problem. The rank decisions may be impeded by these inaccuracies, which may lead to incorrect dimensions for the basis and as a consequence an incorrect, inconsistent set of equations. As such, depending on the equations at hand, the algorithm may still succeed. For example, for $m = 4$, four out of five runs still succeed in finding the solutions with as residuals $0.13 \times 10^{-9}, 0.1587 \times 10^{-7}, 0.29 \times 10^{-9}$, and 0.06×10^{-9} in an average run time of $0.8147s$. This indicates that the algorithm numerical stability issues, severely limiting its practical usefulness and should be fixed before a more elaborate comparison to existing methods is in order.

Try higher precision computing to verify this claim? Also: look into townsends paper

or work in higher accuracy. But then just use Border/Groebner..

6 Conclusion

A novel algorithm for numerical multivariate polynomial root finding was presented. In contrast with existing approaches, the number of basis polynomials that are added in each iteration is decided using the rank of one of the coefficient matrices, rather than symbolically. This eliminates the need to construct the underlying polynomials, relying solely on numerical linear algebra manipulations on the coefficient matrices instead. The algorithm does not consider any explicity ordering or grading on the polynomials either.

Ryzen 7 PRO 5850U with Radeon Graphics, 1901 Mhz, 8 Core and 16 Logical Processors and 32 GB of RAM.

[7]Note that we did not attempt to push the MacaulayLab implementation to failure in terms of model complexity.

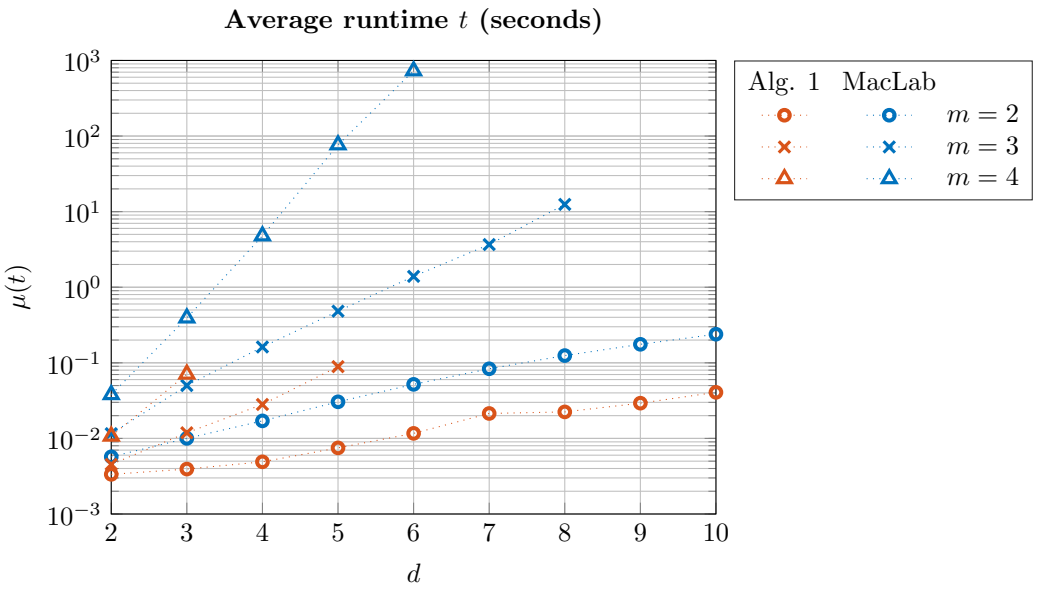

Figure 2: Comparison of the the average runtime over five runs of solving a random system of m equations of degree d in m variables. Experiments ran onto degrees $10, 8, 6, 4$ for $m = 2, 3, 4, 5$, respectively. Missing data indicates that the algorithm failed to find all solutions on at least one of its runs.

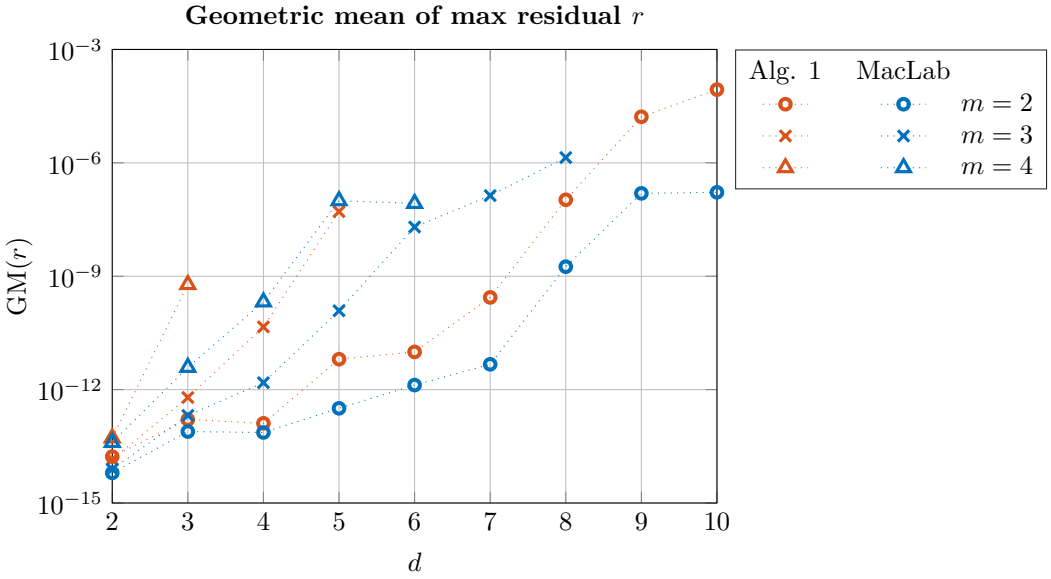

Figure 3: Comparison of the geometric mean of the residuals of five runs of solving a random system of m equations of degree d in m variables. Experiments ran onto degrees $10, 8, 6, 4$ for $m = 2, 3, 4, 5$, respectively. Missing data indicates that the algorithm failed to find all solutions on at least one of its runs.

Even though numerical experiments suggest that the method converges for systems with a finite number of projective roots, a formal convergence proof has yet to be established. Moreover, while the abovementioned properties allow for great flexibility in the choice of underlying polynomials, numerical experiments showed that the stability of the proposed method in its current state seems lackluster at best, hindering any practical use. Nonetheless, we hope that this flexibility and reliance on the well-established methods of linear algebra provides fertile grounds for more numerically stable developments in the future. Better choices of bases may exist, for example, as well as more intricate choices for the new basis elements added in each iteration. Accurately answering these questions requires a more in-depth analysis of the algorithm's stability, however.

References

[1] K. Batselier, P. Dreesen, and B. De Moor. "On the null spaces of the Macaulay matrix". In: *Linear Algebra and its Applications* 460 (2014), pp. 259–289.

[2] Matías R. Bender and Simon Telen. *Yet another eigenvalue algorithm for solving polynomial systems*. 2022. arXiv: `2105.08472`. URL: `https://arxiv.org/abs/2105.08472`.

[3] P. Breiding and S. Timme. "HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia: 6th International Conference, South Bend, IN, USA, July 24-27, 2018, Proc." In: 2018, pp. 458–465.

[4] Robert M. Corless, Patrizia M. Gianni, and Barry M. Trager. "A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots". In: *Proc. of the International Symposium on Symbolic and Algebraic Computation*. 1997, pp. 133–140.

[5] D A Cox, J B Little, and D O'Shea. *Ideals, Varieties and Algorithms*. Springer-Verlag, 2007.

[6] D. A. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. 2nd ed. Springer, 2005.

[7] P. Dreesen, K. Batselier, and B. De Moor. "Multidimensional realisation theory and polynomial system solving". In: *International Journal of Control* 91.12 (2018), pp. 2692–2704.

[8] Ioannis Z. Emiris and Bernard Mourrain. "Matrices in Elimination Theory". In: *Journal of Symbolic Computation* 28.1 (1999), pp. 3–44.

[9] Israel M. Gelfand, Mikhail M. Kapranov, and Andrei V. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Boston, MA: Birkhäuser, 1994.

[10] Achim Kehrein and Martin Kreuzer. "Characterizations of border bases". In: *Journal of Pure and Applied Algebra* 196.2 (2005), pp. 251–270.

[11] F S Macaulay. *The algebraic theory of modular systems*. Cambridge University Press, 1916.

[12] B Mourrain. "A new criterion for normal form algorithms". In: *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. AAECC 1999* (1999), pp. 430–443.

[13] Bernard Mourrain and Philippe Trebuchet. "Generalized normal forms and polynomial system solving". In: *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation*. ISSAC '05. Beijing, China: Association for Computing Machinery, 2005, pp. 253–260.

[14] Bernard Mourrain and Philippe Trébuchet. "Border basis representation of a general quotient algebra". In: *Proc. of the 37th Interantional Symposium on Symbolic and Algebraic Computation*. 2012, pp. 265–272.

[15] Bernard Mourrain and Philippe Trébuchet. "Border basis representation of a general quotient algebra". In: *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*. ISSAC '12. Grenoble, France: Association for Computing Machinery, 2012, pp. 265–272.

[16] Bernard Mourrain and Philippe Trébuchet. "Stable normal forms for polynomial system solving". In: *Theoretical Computer Science* 409.2 (2008). Symbolic-Numerical Computations, pp. 229–240.

[17] H. J. Stetter. *Numerical Polynomial Algebra*. SIAM, 2004, p. 57.

[18] Simon Telen. "Solving Systems of Polynomial Equations". PhD thesis. KU Leuven, 2020.

[19] Simon Telen, Bernard Mourrain, and Marc Van Barel. "Solving Polynomial Systems via Truncated Normal Forms". In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (2018), pp. 1421–1447.

[20] Simon Telen and Marc Van Barel. "A stabilized normal form algorithm for generic systems of polynomial equations". In: *Journal of Computational and Applied Mathematics* 342 (2018), pp. 119–132.

[21] Philippe Trébuchet, Bernard Mourrain, and Marta Abril Bucero. "Border Basis for Polynomial System Solving and Optimization". In: *ICMS 2016*. 5th International Conference on Mathematical Software. Springer, 2016, pp. 212–220.

[22] Christof Vermeersch. "The (Block) Macaulay Matrix: Solving Systems of Multivariate Polynomial Equations and Multiparameter Eigenvalue Problems". eng. PhD thesis. KU Leuven, 2023.

[23] Jan Verschelde. "Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation". In: *ACM Transactions on Mathematical Software* 25.2 (1999), pp. 251–276.