Are resultant methods numerically unstable for multidimensional rootfinding?

Noferini, Vanni and Townsend, Alex

2015

MIMS EPrint: **2015.51**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

ARE RESULTANT METHODS NUMERICALLY UNSTABLE FOR MULTIDIMENSIONAL ROOTFINDING?

VANNI NOFERINI* AND ALEX TOWNSEND†

Abstract. Hidden-variable resultant methods are a class of algorithms for solving multidimensional polynomial rootfinding problems. In two dimensions, when significant care is taken, they are competitive practical rootfinders. However, in higher dimensions they are known to miss zeros, calculate roots to low precision, and introduce spurious solutions. We show that the hidden-variable resultant method based on the Cayley (Dixon or Bézout) resultant is inherently and spectacularly numerically unstable by a factor that grows exponentially with the dimension. We also show that the Sylvester resultant for solving bivariate polynomial systems can square the condition number of the problem. In other words, two popular hidden-variable resultant methods are numerically unstable, and this mathematically explains the difficulties that are frequently reported by practitioners. Along the way, we prove that the Cayley resultant is a generalization of Cramer's rule for solving linear systems and generalize Clenshaw's algorithm to an evaluation scheme for polynomials expressed in a degree-graded polynomial basis.

Key words. resultants, rootfinding, conditioning, multivariate polynomials, Cayley, Sylvester

AMS subject classifications. 13P15, 65H04, 65F35

1. Introduction. Hidden-variable resultant methods are a popular class of algorithms for global multidimensional rootfinding [1, 17, 27, 35, 39, 40]. They compute all the solutions to zero-dimensional polynomial systems of the form:

$$\begin{pmatrix} p_1(x_1,\ldots,x_d) \\ \vdots \\ p_d(x_1,\ldots,x_d) \end{pmatrix} = 0, \qquad (x_1,\ldots,x_d) \in \mathbb{C}^d, \tag{1.1}$$

where $d \geq 2$ and $p_1, \ldots, p_d$ are polynomials in $x_1, \ldots, x_d$ with complex coefficients. Mathematically, they are based on an elegant idea that converts the multidimensional rootfinding problem in (1.1) into one or more eigenvalue problems [6]. At first these methods appear to be a practitioner's dream as a difficult rootfinding problem is solved by the robust QR or QZ algorithm. Desirably, these methods have received considerable research attention from the scientific computing community [10, 18, 30, 46].

Despite this significant interest, hidden-variable resultant methods are notoriously difficult, if not impossible, to make numerically robust. Most naive implementations will introduce unwanted spurious solutions, compute roots inaccurately, and unpredictably miss zeros [8]. Spurious solutions can be removed by manually checking that all the solutions satisfy (1.1), inaccurate roots can usually be polished by Newton's method, but entirely missing a zero is detrimental to a global rootfinding algorithm.

The higher the polynomial degree n and the dimension d, the more pronounced the numerical difficulties become. When $d = 2$ and real finite solutions are of interest, a careful combination of domain subdivision, regularization, and local refinement has been successfully used together with the Cayley resultant (also known as the Dixon or Bézout resultant) for large n [35]. This is the algorithm employed by Chebfun for

*School of Mathematics, University of Manchester, Manchester, UK, M13 9PL. (vanni.noferini@manchester.ac.uk)

†Department of Mathematics, Massachusetts Institute of Technology, 77 Massachusetts Avenue Cambridge, MA 02139-4307. (ajt@mit.edu)

bivariate global rootfinding [45]. Moreover, for $d = 2$, randomization techniques and the QZ algorithm have been combined fruitfully with the Macaulay resultant [27]. There are also many other ideas [4, 33]. However, these techniques seem to be less successful in higher dimensions.

In this paper, we show that any plain vanilla hidden-variable resultant method based on the Cayley or Sylvester resultant matrix is a numerically unstable algorithm for solving a polynomial system. In particular, we show that the hidden-variable resultant method based on the Cayley resultant matrix is numerically unstable for multidimensional rootfinding with a factor that grows exponentially with d. We show that for $d = 2$ the Sylvester resultant matrix leads to a hidden-variable resultant method that can also square the conditioning of a root.

We believe that this numerical instability has not been analyzed before because there are at least two other sources of numerical issues: (1) The hidden-variable resultant method is usually employed with the monomial polynomial basis, which can be devastating in practice when n is large, and (2) Some rootfinding problems have inherently ill-conditioned zeros and hence, one does not always expect accurate solutions. Practitioners can sometimes overcome (1) by representing the polynomials $p_1, \ldots, p_d$ in another degree-graded polynomial basis[1] such as the Chebyshev or Legendre polynomial basis [8]. However, the numerically instability that we identify can be observed even when the solutions are well-conditioned and for degree-graded polynomial basis.

We focus on the purely numerical, as opposed to symbolic, algorithm. We take the view that every arithmetic operation is performed in finite precision. There are many other rootfinders that either employ only symbolic manipulations [9] or some kind of symbolic-numerical hybrid [19]. Similar careful symbolic manipulations may be useful in overcoming the numerical instability that we identify. For example, it may be possible to somehow transform the polynomial system (1.1) into one that the resultant method treats in a numerical stable manner.

This paper may be considered as a bearer of bad news. Yet, we take the opposite and more optimistic view. We are intrigued by the potential positive impact this paper could have on rootfinders based on resultants since once a numerical instability has been identified the community is much better placed to circumvent the issue.

Our setup is as follows. First, we suppose that a degree-graded polynomial basis for $\mathbb{C}_n[x]$, denoted by $\phi_0, \ldots, \phi_n$, has been selected. All polynomials will be represented using this basis. Second, a region of interest $\Omega^d \subset \mathbb{C}^d$ is chosen such that Ω^d, where Ω^d is the tensor-product domain $\Omega \times \cdots \times \Omega$ (d times), contains all the roots that would like to be computed accurately. The domain $\Omega \subset \mathbb{C}$ can be a real interval or a bounded region in the complex plane. Throughout, we suppose that $\sup_{x\in\Omega} |\phi_k(x)| = 1$ for $0 \le k \le n$, which is a very natural normalization.

We use the following notation. The space of univariate polynomials with complex coefficients of degree at most n is denoted by $\mathbb{C}_n[x]$, the space of d-variate polynomials of maximal degree n in the variables $x_1, \ldots, x_d$ is denoted by $\mathbb{C}_n[x_1, \ldots, x_d]$, and if $\mathcal{V}$ is a vector space then the Cartesian product space $\mathcal{V} \times \cdots \times \mathcal{V}$ (d-times) is denoted by $(\mathcal{V})^d$. Finally, we use vec(V) to be the vectorization of the matrix or tensor V to a column vector (this is equivalent to `V(:)` in MATLAB).

Our two main results are in theorems 3.7 and 4.6. Together they show that there

[1] A polynomial basis $\{\phi_0, \ldots, \phi_n\}$ for $\mathbb{C}_n[x]$ is degree-graded if $\phi_k(x)$ is of degree exactly k for $0 \le k \le n$.

exist $p_1, \ldots, p_d$ in (1.1) such that

$$\underbrace{\kappa(x_d^*, R)}_{\text{Cond. no. of the eigenproblem}} \geq (\underbrace{\|J(\underline{x}^*)^{-1}\|_2}_{\text{Cond. no. of } \underline{x}^*})^d,$$

where R is either the Cayley (for any $d \geq 2$) or Sylvester (for $d = 2$) resultant matrix. Such a result shows that in the absolute sense the eigenvalue problem employed by these two resultant-based methods can be significantly more sensitive to perturbations than the corresponding root. Together with results about relative conditioning, we conclude that these rootfinders are numerically unstable (see Section 5).

In the next section we first introduce multidimensional resultants and describe hidden-variable resultant methods for rootfinding. In Section 3 we show that the hidden-variable resultant method based on the Cayley resultant suffers from numerical instability and in Section 4 we show that the Sylvester resultant matrix has a similar instability for $d = 2$. In Section 5 we explain why our absolute conditioning analysis leads to an additional twist when considering relative conditioning. Finally, in Section 6 we present a brief outlook on future directions.

2. Background material. This paper requires some knowledge of multidimensional rootfinding, hidden-variable resultant methods, matrix polynomials, and conditioning analysis. In this section we briefly review this material.

2.1. Global multidimensional rootfinding. Global multidimensional rootfinding can be a difficult and computationally expensive task. Here, we are concerned with the easiest situation where (1.1) has only simple finite roots.

DEFINITION 2.1 (Simple root). *Let $\underline{x}^* = (x_1^*, \ldots, x_d^*) \in \mathbb{C}^d$ be a solution to the zero-dimensional polynomial system* (1.1)*. Then, we say that $\underline{x}^*$ is a simple root of* (1.1) *if the Jacobian matrix $J(\underline{x}^*)$ is invertible, where*

$$J(\underline{x}^*) = \begin{bmatrix} \frac{\partial p_1}{\partial x_1}(\underline{x}^*) & \cdots & \frac{\partial p_1}{\partial x_d}(\underline{x}^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial p_d}{\partial x_1}(\underline{x}^*) & \cdots & \frac{\partial p_d}{\partial x_d}(\underline{x}^*) \end{bmatrix} \in \mathbb{C}^{d \times d}. \tag{2.1}$$

If $J(\underline{x}^*)$ is not invertible then the problem is ill-conditioned, and a numerically stable algorithm working in finite precision arithmetic may introduce a spurious solution or may miss a non-simple root entirely. We will consider the roots of (1.1) that are well-conditioned (see Proposition 2.9), finite, and simple.

Our primary focus is on the accuracy of hidden-variable resultant methods, not computational speed. In general, one cannot expect to have a "fast" algorithm for global multidimensional rootfinding. This is because the zero-dimensional polynomial system in (1.1) can potentially have a large number of solutions. To say exactly how many solutions there can be, we first must be more precise about what we mean by the degree of a polynomial in the multidimensional setting [38].

DEFINITION 2.2 (Polynomial degree). *A d-variate polynomial $p(x_1, \ldots, x_d)$ has total degree $\leq n$ if*

$$p(x_1, \ldots, x_d) = \sum_{i_1 + \cdots + i_d \leq n} A_{i_1, \ldots, i_d} \prod_{k=1}^{d} \phi_{i_k}(x_k)$$

for some tensor A. It is of total degree n if one of the terms $A_{i_1,\ldots,i_d}$ with $i_1+\cdots+i_d = n$ is nonzero. Moreover, $p(x_1,\ldots,x_d)$ has maximal degree $\leq n$ if

$$p(x_1,\ldots,x_d) = \sum_{i_1,\cdots,i_d=0}^{n} A_{i_1,\ldots,i_d} \prod_{k=1}^{d} \phi_{i_k}(x_k)$$

for some tensor A indexed by $0 \leq i_1,\ldots,i_d \leq n$. It is of maximal degree n if one of the terms $A_{i_1,\ldots,i_d}$ with $\max(i_1,\ldots,i_d) = n$ is nonzero.

Bézout's Lemma says that if (1.1) involves polynomials of total degree n, then there are at most n^d solutions [29, Chap. 3]. For polynomials of maximal degree we have the following analogous bound (see also [44, Thm. 5.1]).

LEMMA 2.3. *The zero-dimensional polynomial system in* (1.1)*, where $p_1,\ldots,p_d$ are of maximal degree n, can have at most $d!n^d$ solutions.*

Proof. By [38, Thm. 8.5.2], the polynomial system (1.1) can have no more solutions than the so-called mixed volume bound. For polynomials of maximal degree this bound can be calculated as follows:

$$\sum_{k=1}^{d}(-1)^{d-k}\binom{d}{k}(kn)^d = n^d\sum_{k=1}^{d}(-1)^{d-k}\binom{d}{k}k^d = d!n^d.$$

□

We have selected maximal degree, rather than total degree, because maximal degree polynomials are more closely linked to tensor-product constructions and hence, are easier to work with numerically in the multidimensional setting.

Suppose that the polynomial system (1.1) contains polynomials of maximal degree n. Then, to verify that $d!n^d$ candidate points are solutions the polynomials $p_1,\ldots,p_d$ must be evaluated, costing $\mathcal{O}(n^{2d})$ operations. Thus, the optimal worst-case complexity is $\mathcal{O}(n^{2d})$. For many applications global rootfinding is computationally unfeasible and instead local methods such as Newton's method and homotopy continuation methods [3] can be employed to compute a subset of the solutions. Despite the fact that global multidimensional rootfinding is a computationally intensive task, we still desire a numerically stable algorithm. A survey of numerical rootfinders is given in [44, Chap. 5].

When $d = 1$, global numerical rootfinding can be done satisfactorily even with polynomial degrees in the thousands. Excellent numerical and stable rootfinders can be built using domain subdivision [7], eigenproblems with colleague or comrade matrices [23], and a careful treatment of dynamic range issues [7].

2.2. Hidden-variable resultant methods. The first step of a hidden-variable resultant method is to select a variable, say x_d, and regard the d-variate polynomials $p_1,\ldots,p_d$ in (1.1) as polynomials in $x_1,\ldots,x_{d-1}$ with complex coefficients that depend on x_d. That is, we "hide" x_d by rewriting $p_k(x_1,\ldots,x_d)$ for $1 \leq k \leq d$ as

$$p_k(x_1,\ldots,x_{d-1},x_d) = p_k[x_d](x_1,\ldots,x_{d-1}) = \sum_{i_1,\cdots,i_{d-1}=0}^{n} c_{i_1,\ldots,i_{d-1}}(x_d)\prod_{s=1}^{d-1}\phi_{i_s}(x_s),$$

where $\{\phi_0,\ldots,\phi_n\}$ is a degree-graded polynomial basis for $\mathbb{C}_n[x]$. This new point-of-view rewrites (1.1) as a system of d polynomials in $d-1$ variables. We now seek all the $x_d^* \in \mathbb{C}$ such that $p_1[x_d^*],\ldots,p_d[x_d^*]$ have a common root in Ω^{d-1}. Algebraically, this can be achieved by using a multidimensional resultant [20, Chap. 13].

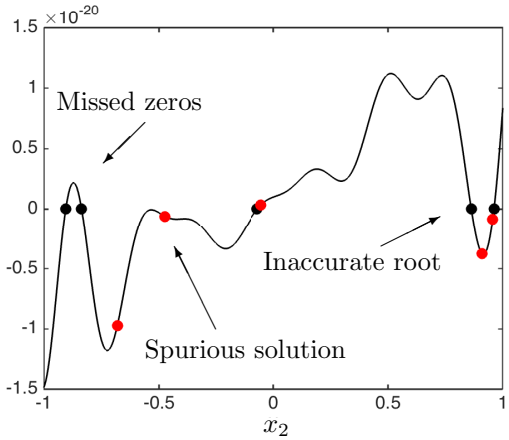

FIG. 1. *Mathematically, the zeros of $\mathcal{R}(p_1[x_d], \ldots, p_d[x_d])$ are the dth component of the solutions to (1.1). However, numerically the polynomial $\mathcal{R}(p_1[x_d], \ldots, p_d[x_d])$ can be numerically close to zero everywhere. Here, we depict the typical behavior of the polynomial $\mathcal{R}(p_1[x_d], \ldots, p_d[x_d])$ when $d = 2$, where the black dots are the exact zeros and the red dots are the computed roots.*

DEFINITION 2.4 (Multidimensional resultant). *Let $d \geq 2$ and $n \geq 0$. A functional $\mathcal{R} : (\mathbb{C}_n[x_1, \ldots, x_{d-1}])^d \to \mathbb{C}$ is a multidimensional resultant if, for any set of d polynomials $q_1, \ldots, q_d \in \mathbb{C}_n[x_1, \ldots, x_{d-1}]$, $\mathcal{R}(q_1, \ldots, q_d)$ is a polynomial in the coefficients of $q_1, \ldots, q_d$ and $\mathcal{R}(q_1, \ldots, q_d) = 0$ if and only if there exists an $\underline{x}^* \in \tilde{\mathbb{C}}^{d-1}$ such that $q_k(\underline{x}^*) = 0$ for $1 \leq k \leq d$, where $\tilde{\mathbb{C}}$ denotes the extended complex plane*[2].

Definition 2.4 makes $\mathcal{R}$ unique up to a multiplicative nonzero constant [11, Thm. 1.6.1(i)]. Moreover, in the monomial basis it is standard to normalize $\mathcal{R}$ so that $\mathcal{R}(x_1^n, \ldots, x_{d-1}^n, 1) = 1$ [11, Thm. 1.6.1(ii)]. For nonmonomial bases, we are not aware of any standard normalization.

Assuming (1.1) only has finite solutions, if $\mathcal{R}$ is a multidimensional resultant then for any $x_d^* \in \mathbb{C}$ we have

$$\mathcal{R}(p_1[x_d^*], \ldots, p_d[x_d^*]) = 0 \iff \exists (x_1^*, \ldots, x_{d-1}^*) \in \mathbb{C}^{d-1} \text{ s.t. } p_1(\underline{x}^*) = \cdots = p_d(\underline{x}^*) = 0,$$

where $\underline{x}^* = (x_1^*, \ldots, x_d^*) \in \mathbb{C}^d$. Thus, we can calculate the dth component of all the solutions of interest by computing the roots of $\mathcal{R}(p_1[x_d], \ldots, p_d[x_d])$ and discarding those outside of Ω. In principle, since $\mathcal{R}(p_1[x_d], \ldots, p_d[x_d])$ is a univariate polynomial in x_d it is an easy task. However, numerically, $\mathcal{R}$ is typically near-zero in large regions of $\mathbb{C}$, and spurious solutions as well as missed zeros plague this approach in finite precision arithmetic (see Figure 1). Thus, directly computing the roots of $\mathcal{R}$ is spectacularly numerically unstable for almost all n and d. This approach is rarely advocated in practice.

Instead, one often considers an associated multidimensional resultant matrix whose determinant is equal to $\mathcal{R}$. Working with matrices rather than determinants is beneficial for practical computations, especially when $d = 2$ [17, 35, 39]. Occasionally, this variation on hidden-variable resultant methods is called *numerically confirmed eliminants* to highlight its improved numerical behavior [38, Sec. 6.2.2]. However, we will show that even after this significant improvement the hidden-variable resultant methods based on the Cayley and Sylvester resultant matrices remain numerically unstable.

[2]To make sense of solutions at infinity one can work with homogeneous polynomials [11].

DEFINITION 2.5 (Multidimensional resultant matrix). *Let $d \geq 2$, $n \geq 0$, $N \geq 1$, and $\mathcal{R}$ a multidimensional resultant. A matrix-valued function $R : (\mathbb{C}_n[x_1, \ldots, x_{d-1}])^d \to \mathbb{C}^{N \times N}$ is a multidimensional resultant matrix associated with $\mathcal{R}$ if for any set of d polynomials $q_1, \ldots, q_d \in \mathbb{C}_n[x_1, \ldots, x_{d-1}]$ we have*

$$\det\left(R(q_1, \ldots, q_d)\right) = \mathcal{R}(q_1, \ldots, q_d).$$

There are many types of resultant matrices including Cayley (see Section 3), Sylvester (see Section 4), Macaulay [27], and others [18, 28, 32]. In this paper we only consider two of the most popular choices that are the Cayley and Sylvester resultant matrices.

Theoretically, we can calculate the dth component of the solutions by finding all the $x_d^* \in \mathbb{C}$ such that $\det(R(p_1[x_d^*], \ldots, p_d[x_d^*])) = 0$. In practice, our analysis will show that this dth component cannot always be accurately computed.

Each entry of the matrix $R(p_1[x_d], \ldots, p_d[x_d])$ is a polynomial in x_d of finite degree. In linear algebra such objects are called matrix polynomials (or polynomial matrices) and finding the solutions of $\det(R(p_1[x_d], \ldots, p_d[x_d])) = 0$ is related to a polynomial eigenproblem [5, 31, 43].

2.3. Matrix polynomials. Since multidimensional resultant matrices are matrices with univariate polynomial entries, matrix polynomials play an important role in the hidden-variable resultant method. A classical reference on matrix polynomials is the book by Gohberg, Lancaster, and Rodman [22].

DEFINITION 2.6 (Matrix polynomial). *Let $N \geq 1$ and $K \geq 0$. We say that $P(\lambda)$ is a (square) matrix polynomial of size N and degree K if $P(\lambda)$ is an $N \times N$ matrix whose entries are univariate polynomials in λ of degree $\leq K$, where at least one entry is of degree exactly K.*

In fact, since (1.1) is a zero-dimensional polynomial system it can only have a finite number of isolated solutions and hence, the matrix polynomials we consider are regular [22].

DEFINITION 2.7 (Regular matrix polynomial). *We say that a square matrix polynomial $P(\lambda)$ is regular if $\det(P(\lambda)) \neq 0$ for some $\lambda \in \mathbb{C}$.*

A matrix polynomial $P(\lambda)$ of size N and degree K can be expressed in a degree-graded polynomial basis as

$$P(\lambda) = \sum_{i=0}^{K} A_i \phi_i(\lambda), \qquad A_i \in \mathbb{C}^{N \times N}. \tag{2.2}$$

When the leading coefficient matrix A_K in (2.2) is invertible the eigenvalues of $P(\lambda)$ are all finite, and they satisfy $\det(P(\lambda)) = 0$.

DEFINITION 2.8 (Eigenvector of a regular matrix polynomial). *Let $P(\lambda)$ be a regular matrix polynomial of size N and degree K. If $\lambda \in \mathbb{C}$ is finite and there exists a non-zero vector $v \in \mathbb{C}^{N \times 1}$ such that $P(\lambda)v = 0$ (resp. $v^T P(\lambda) = 0$), then we say that v is a* right (resp. left) eigenvector *of $P(\lambda)$ corresponding to the eigenvalue λ.*

For a regular matrix polynomial $P(\lambda)$ we have the following relationship between its eigenvectors and determinant [22]: For any finite $\lambda \in \mathbb{C}$,

$$\det(P(\lambda)) = 0 \quad \Longleftrightarrow \quad \exists v \in \mathbb{C}^{N \times 1} \setminus \{0\}, \quad P(\lambda)v = 0.$$

In multidimensional rootfinding, one sets $P(\lambda) = R(p_1[\lambda], \ldots, p_d[\lambda])$ and solves $\det(P(\lambda)) = 0$ via the polynomial eigenvalue problem $P(\lambda)v = 0$. There are various

algorithms for solving $P(\lambda)v = 0$ including linearization [22, 31, 43], the Ehrlich–Aberth method [5, 21, 41], and contour integration [2]. However, regardless of how the polynomial eigenvalue problem is solved in finite precision, the hidden-variable resultant method based on the Cayley or the Sylvester resultant matrix is numerically unstable.

For the popular resultant matrices, such as Cayley and Sylvester, the first $d-1$ components of the solutions can be determined from the left or right eigenvectors of $R(p_1[x_d^*], \ldots, p_d[x_d^*])$. For instance, if linearization is employed, the multidimensional rootfinding problem is converted into one (typically very large) eigenproblem, which can be solved by the QR or QZ algorithm. Practitioners often find that the computed eigenvectors are not accurate enough to adequately determine the $d-1$ components. However, the blame for the observed numerical instability is not only on the eigenvectors, but also the eigenvalues. Our analysis will show that the dth component may not be computed accurately either.

2.4. Conditioning analysis. Not even a numerically stable algorithm can be expected to accurately compute a simple root of (1.1) if that root is itself sensitive to small perturbations. Finite precision arithmetic almost always introduces roundoff errors and if these can cause large perturbations in a root then that solution is ill-conditioned.

The absolute condition number of a simple root measures how sensitive the location of the root is to small perturbations in $p_1, \ldots, p_d$.

PROPOSITION 2.9 (The absolute condition number of a simple root). *Let $\underline{x}^* = (x_1^*, \ldots, x_d^*) \in \mathbb{C}^d$ be a simple root of* (1.1). *The absolute condition number of $\underline{x}^*$ associated with rootfinding is $\|J(\underline{x}^*)^{-1}\|_2$, i.e., the matrix 2-norm of the inverse of the Jacobian.*

Proof. See [35]. □

As a rule of thumb, a numerically stable rootfinder should be able to compute a simple root $\underline{x}^* \in \mathbb{C}^d$ to an accuracy of $\mathcal{O}(\max(\|J(\underline{x}^*)^{-1}\|_2, 1)u)$, where u is the unit machine roundoff. In contrast, regardless of the condition number of $\underline{x}^*$, a numerically unstable rootfinder may not compute it accurately. Worse still, it may miss solutions with detrimental consequences.

A hidden-variable resultant method computes the dth component of the solutions by solving the polynomial eigenvalue problem $R(p_1[x_d], \ldots, p_d[x_d])v = 0$. The following condition number tells us how sensitive an eigenvalue is to small perturbations in R [35, (12)] (also see [42]):

DEFINITION 2.10 (The absolute condition number of an eigenvalue of a regular matrix polynomial). *Let $x_d^* \in \mathbb{C}$ be a finite eigenvalue of $R(x_d) = R(p_1[x_d], \ldots, p_d[x_d])$. The condition number of x_d^* associated with the eigenvalue problem $R(x_d)v = 0$ is*

$$\kappa(x_d^*, R) = \lim_{\epsilon \to 0^+} \sup \left\{ \frac{1}{\epsilon} \min |\hat{x}_d - x_d^*| : \det(\widehat{R}(\hat{x}_d)) = 0 \right\}, \tag{2.3}$$

where the supremum is taken over the set of matrix polynomials $\widehat{R}(x_d)$ such that $\max_{x_d \in \Omega} \|\widehat{R}(x_d) - R(x_d)\|_2 \leq \epsilon$.

A numerical polynomial eigensolver can only be expected to compute the eigenvalue x_d^* satisfying $R(x_d^*)v = 0$ to an accuracy of $\mathcal{O}(\max(\kappa(x_d^*, R), 1)u)$, where u is unit machine roundoff. We will be interested in how $\kappa(x_d^*, R)$ relates to the condition number, $\|J(\underline{x}^*)^{-1}\|_2$, of the corresponding root.

It can be quite difficult to calculate $\kappa(x_d^*, R)$ directly from (2.3), and is usually more convenient to use the formula below. (Related formulas can be found in [35,

Thm. 1] for symmetric matrix polynomials and in [42, Thm. 5] for general matrix polynomials.)

LEMMA 2.11. *Let $R(x_d)$ be a regular matrix polynomial with finite simple eigenvalues. Let $x_d^* \in \mathbb{C}$ be an eigenvalue of $R(x_d)$ with corresponding right and left eigenvectors $v, w \in \mathbb{C}^{N\times 1}$. Then, we have*

$$\kappa(x_d^*, R) = \frac{\|v\|_2\|w\|_2}{|w^T R'(x_d)v|},$$

where $R'(x_d)$ denotes the derivative of R with respect to x_d.

Proof. The first part of the proof follows the analysis in [42]. Let $R(x_d)$ be a regular matrix polynomial with a simple eigenvalue $x_d^* \in \mathbb{C}$ and corresponding right and left eigenvectors $v, w \in \mathbb{C}^{N\times 1}$. A perturbed matrix polynomial $\hat{R}(x) = R(x) + \Delta R(x)$ will have a perturbed eigenvalue $\hat{x}_d$ and a perturbed eigenvector $\hat{v} = v + \delta v$ such that $R(\hat{x}_d)\hat{v} + \Delta R(\hat{x}_d)\hat{v} = 0$, where $\|\Delta R(x)\|_2 \leq \epsilon$.

Expanding, keeping only the first order terms, and using $R(x_d^*)v = 0$ we obtain

$$(\hat{x}_d - x_d^*)R'(x_d^*)v + R(x_d^*)\delta v + \Delta R(x_d^*)v = \mathcal{O}(\epsilon^2).$$

Multiplying by w^T on the left, rearranging, and keeping the first order terms, we obtain

$$\hat{x}_d = x_d^* - \frac{w^T \Delta R(x_d^*)v}{w^T R'(x_d^*)v},$$

where the derivative in $R'(x_d^*)$ is taken with respect to x_d. Thus, from (2.3) we see that

$$\kappa(x_d^*, R) \leq \frac{\|v\|_2\|w\|_2}{|w^T R'(x_d^*)v|}. \tag{2.4}$$

We now show that the upper bound in (2.4) can be attained. Take $\Delta R(x_d) = \epsilon w v^T/(\|v\|_2\|w\|_2)$. Then, $\max_{x_d\in\Omega} \|\Delta R(x_d)\|_2 = \epsilon$ and

$$\frac{w^T \Delta R(x_d^*)v}{w^T R'(x_d^*)v} = \epsilon \frac{\|v\|_2\|w\|_2}{w^T R'(x_d^*)v}.$$

The result follows by Definition 2.10. □

For the Cayley resultant matrix (see Section 3), we will show that $\kappa_2(x_d^*, R)$ can be as large as $\|J(\underline{x}^*)^{-1}\|_2^d$ (see Theorem 3.7). Thus, there can be an exponential increase in the conditioning that seems inherent to the methodology of the hidden-variable resultant method based on the Cayley resultant matrix. In particular, once the polynomial eigenvalue problem has been constructed, a backward stable numerical eigensolver may not compute accurate solutions to (1.1).

We now must tackle the significant challenge of showing that the Cayley and Sylvester resultant matrices do lead to numerical unstable hidden-variable resultant methods, i.e., for certain solutions $\underline{x}^*$ the quantity $\kappa_2(x_d^*, R)$ can be much larger than $\|J(\underline{x}^*)^{-1}\|_2$.

3. The Cayley resultant is numerically unstable for multidimensional rootfinding. The hidden-variable resultant method based on the Cayley resultant [12] finds the solutions to (1.1) by solving the polynomial eigenvalue problem $R_{Cayley}(x_d)v =$

0, where $R_{Cayley}(x_d)$ is a certain matrix polynomial. To define it we follow the exposition in [13] and first introduce a related Cayley function f_{Cayley}.

DEFINITION 3.1 (Cayley function). *The Cayley function associated with* $q_1, \dots, q_d \in \mathbb{C}_n[x_1, \dots, x_{d-1}]$ *is a multivariate polynomial in* $2d-2$ *variables, denoted by* $f_{Cayley} = f_{Cayley}(q_1, \dots, q_d)$*, and is given by*

$$f_{Cayley} = \det \begin{pmatrix} q_1(s_1, s_2, \dots, s_{d-1}) & \dots & q_d(s_1, s_2, \dots, s_{d-1}) \\ q_1(t_1, s_2, \dots, s_{d-1}) & \dots & q_d(t_1, s_2, \dots, s_{d-1}) \\ \vdots & \ddots & \vdots \\ q_1(t_1, t_2, \dots, t_{d-1}) & \dots & q_d(t_1, t_2, \dots, t_{d-1}) \end{pmatrix} \Bigg/ \prod_{i=1}^{d-1} (s_i - t_i). \quad (3.1)$$

In two dimensions the Cayley function (also known as the Bézoutian function [34]) takes the more familiar form of

$$f_{Cayley} = \frac{1}{s_1 - t_1} \det \begin{pmatrix} q_1(s_1) & q_2(s_1) \\ q_1(t_1) & q_2(t_1) \end{pmatrix} = \frac{q_1(s_1)q_2(t_1) - q_2(s_1)q_1(t_1)}{s_1 - t_1},$$

which is of degree at most $n-1$ in s_1 and t_1. By carefully applying Laplace's formula for the matrix determinant in (3.1), one can see that f_{Cayley} is a polynomial of degree $\tau_k \le kn - 1$ in s_k and t_{d-k} for $1 \le k \le d-1$.

Note that f_{Cayley} is not the multidimensional resultant (except when $\tau_k = 0$ for all k). Instead, f_{Cayley} is a function that is a convenient way to define the Cayley resultant matrix.

Let $\{\phi_0, \phi_1, \dots, \}$ be the selected degree-graded polynomial basis. The Cayley resultant matrix depends on the polynomial basis and is related to the expansion coefficients of f_{Cayley} in a tensor-product basis of $\{\phi_0, \phi_1, \dots, \}$. That is, let

$$f_{Cayley} = \sum_{i_1=0}^{\tau_1} \cdots \sum_{i_{d-1}=0}^{\tau_{d-1}} \sum_{j_1=0}^{\tau_{d-1}} \cdots \sum_{j_{d-1}=0}^{\tau_1} A_{i_1,\dots,i_{d-1},j_1,\dots,j_{d-1}} \prod_{k=1}^{d-1} \phi_{i_k}(s_k) \prod_{k=1}^{d-1} \phi_{j_k}(t_k) \quad (3.2)$$

be the tensor-product expansion of the polynomial f_{Cayley}, where A is a tensor of expansion coefficients of size $(\tau_1 + 1) \times \cdots \times (\tau_{d-1} + 1) \times (\tau_{d-1} + 1) \times \cdots \times (\tau_1 + 1)$. The *Cayley resultant matrix* is the following unfolding (or matricization) of A [36, Sec. 2.3]:

DEFINITION 3.2 (Cayley resultant matrix). *The Cayley resultant matrix associated with* $q_1, \dots, q_d \in \mathbb{C}_n[x_1, \dots, x_{d-1}]$ *with respect to the basis* $\{\phi_0, \phi_1, \dots, \}$ *is denoted by* R_{Cayley} *and is the* $\left(\prod_{k=1}^{d-1}(\tau_k + 1)\right) \times \left(\prod_{k=1}^{d-1}(\tau_k + 1)\right)$ *matrix formed by the unfolding of the tensor* A *in* (3.2)*. This unfolding is often denoted by* $A_{\boldsymbol{r} \times \boldsymbol{c}}$*, where* $\boldsymbol{r} = \{1, \dots, d-1\}$ *and* $\boldsymbol{c} = \{d, \dots, 2d-2\}$ [36, Sec. 2.3].

For example, when $\tau_k = kn - 1$ for $1 \le k \le d-1$ we have for $0 \le i_k, j_{d-k} \le kn - 1$

$$R_{Cayley}\left(\sum_{k=1}^{d-1} (k-1)! i_k n^{k-1}, \sum_{k=1}^{d-1} j_{d-k} \frac{(d-1)!}{(d-k)!} n^{k-1}\right) = A_{i_1,\dots,i_{d-1},j_1,\dots,j_{d-1}}.$$

Note that this is equivalent to `N = factorial(d-1)*n^(d-1); R = reshape(A,N,N);` in MATLAB, except here the indexing of the matrix R_{Cayley} starts at 0.

For rootfinding, we set $q_1 = p_1[x_d], \dots, q_d = p_d[x_d]$ (thinking of x_d as the "hidden" variable). Then, $R_{Cayley} = R_{Cayley}(x_d)$ is a square matrix polynomial (see

Section 2.3). If all the polynomials are of maximal degree n, then R_{Cayley} is of size $(d-1)!n^{d-1}$ and of degree at most dn. The fact that $(d-1)!n^{d-1} \times dn = d!n^d$ is the maximum number of possible solutions that (1.1) can possess (see Lemma 2.3) is a consequence of R_{Cayley} being a resultant matrix. In particular, the eigenvalues of $R_{Cayley}(x_d)$ are the dth components of the solutions to (1.1) and the remaining $d-1$ components of the solutions can in principle be obtained from the eigenvectors.

It turns out that evaluating f_{Cayley} at $t_1^*, \ldots, t_{d-1}^*$ is equivalent to a matrix-vector product with R_{Cayley}. This relationship between R_{Cayley} and f_{Cayley} will be essential in Section 3.2 for understanding the eigenvectors of R_{Cayley}.

LEMMA 3.3. *Let $d \geq 2$, $\underline{t}^* \in \mathbb{C}^{d-1}$, and f_{Cayley} and R_{Cayley} be the Cayley function and matrix associated with $q_1, \ldots, q_d \in \mathbb{C}_n[x_1, \ldots, x_{d-1}]$, respectively. If V is the tensor satisfying $V_{j_1,\ldots,j_{d-1}} = \prod_{k=1}^{d-1} \phi_{j_k}(t_k^*)$ for $0 \leq j_{d-k} \leq \tau_k$, then we have*

$$R_{Cayley}\text{vec}(V) = \text{vec}(Y),$$

where Y is the tensor that satisfies

$$f_{Cayley}(s_1, \ldots, s_{d-1}, t_1^*, \ldots, t_{d-1}^*) = \sum_{i_1=0}^{\tau_1} \cdots \sum_{i_{d-1}=0}^{\tau_{d-1}} Y_{i_1,\ldots,i_{d-1}} \prod_{k=1}^{d-1} \phi_{i_k}(s_k).$$

Proof. The matrix-vector product $R_{Cayley}\text{vec}(V) = \text{vec}(Y)$ is equivalent to the following sums:

$$\sum_{j_1=0}^{\tau_{d-1}} \cdots \sum_{j_{d-1}=0}^{\tau_1} A_{i_1,\ldots,i_{d-1},j_1,\ldots,j_{d-1}} \prod_{k=1}^{d-1} \phi_{j_k}(t_k^*) = Y_{i_1,\ldots,i_{d-1}}$$

for some tensor Y. The result follows from (3.2). ∎

3.1. The Cayley resultant as a generalization of Cramer's rule. In this section we show that for systems of linear polynomials, i.e., of total degree 1, the Cayley resultant is precisely Cramer's rule. We believe this connection is folklore, but we have been unable to find an existing reference that provides a rigorous justification. It gives a first hint that the hidden-variable resultant method in full generality may be numerically unstable.

THEOREM 3.4. *Let A be a matrix of size $d \times d$, $\underline{x} = (x_1, \ldots, x_d)^T$, and $\underline{b}$ a vector of size $d \times 1$. Then, solving the linear polynomial system $A\underline{x} + \underline{b} = 0$ by the hidden-variable resultant method based on the Cayley resultant is equivalent to Cramer's rule for calculating x_d.*

Proof. Let A_d be the last column of A and $B = A - A_d e_d^T + \underline{b} e_d^T$, where e_d is the dth canonical vector. Recall that Cramer's rule computes the entry x_d in $A\underline{x} = -\underline{b}$ via the formula $x_d = -\det(B)/\det(A)$. We will show that for the linear polynomial system $A\underline{x} + \underline{b} = 0$ we have $f_{Cayley} = \det(B) + x_d \det(A)$. Observe that this, in particular, implies that (since f_{Cayley} has degree 0 in s_i, t_i for all i) $f_{Cayley} = R_{Cayley} = \mathcal{R}_{Cayley}$. Hence, the equivalence between Cramer's rule and rootfinding based on the Cayley resultant.

First, using (3.1), we write $f_{Cayley} = \det(M)/\det(V)$ where the matrices M and

V are

$$V = \begin{bmatrix} s_1 & t_1 & t_1 & \dots & t_1 \\ s_2 & s_2 & t_2 & \dots & t_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{d-1} & s_{d-1} & s_{d-1} & \dots & t_{d-1} \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}, \qquad M = BV + x_d A_d e^T,$$

where e is the $d \times 1$ vector of all ones. (It can be shown by induction on d that $\det(V) = \prod_{i=1}^{d-1}(s_i - t_i)$, as required.) Using the matrix determinant lemma, we have

$$\det(M) = \det(B)\det(V) + x_d e^T \operatorname{adj}(BV) A_d,$$

where $\operatorname{adj}(BV)$ is the algebraic adjugate matrix of BV. Now, recall that $\operatorname{adj}(BV) = \operatorname{adj}(V)\operatorname{adj}(B)$ and observe that $e^T \operatorname{adj}(V) = \det(V) e_d^T$. Hence, we obtain

$$\frac{\det(M)}{\det(V)} = \det(B) + x_d(e_d^T \operatorname{adj}(B) A_d).$$

Using $e_d^T \operatorname{adj}(B)\underline{b} = \det(B)$ and the matrix determinant lemma one more time, we conclude that

$$\det(A) = \det(B) + e_d^T \operatorname{adj}(B) A_d - e_d^T \operatorname{adj}(B)\underline{b} = e_d^T \operatorname{adj}(B) A_d.$$

Thus, $f_{Cayley} = \det(B) + x_d \det(A)$ and the resultant method calculates x_d via Cramer's formula. □

It is well-known in the literature that Cramer's rule is a numerically unstable algorithm for solving $A\underline{x} = \underline{b}$ [24, Sec. 1.10.1]. Thus, Theorem 3.4 casts significant suspicion on the numerical properties of the hidden-variable resultant method based on the Cayley resultant.

3.2. The eigenvector structure of the Cayley resultant matrix. Ultimately, we wish to use Lemma 2.11 to estimate the condition number of the eigenvalues of the Cayley resultant matrix. To do this we need to know the left and right eigenvectors of R_{Cayley}. The following lemma shows that the eigenvectors of R_{Cayley} are in Vandermonde form[3]. To show this we exploit the convenient relationship between evaluation of f_{Cayley} and matrix-vector products with R_{Cayley}.

LEMMA 3.5. *Suppose that $\underline{x}^* = (x_1^*, \dots, x_d^*) \in \mathbb{C}^d$ is a simple root of (1.1). Let V and W be tensors of size $(\tau_{d-1}+1) \times \cdots \times (\tau_1 + 1)$ and $(\tau_1+1) \times \cdots \times (\tau_{d-1}+1)$, respectively, defined by*

$$V_{j_1,\dots,j_{d-1}} = \prod_{k=1}^{d-1} \phi_{j_k}(x_k^*), \qquad 0 \le j_k \le \tau_{d-k}$$

and

$$W_{i_1,\dots,i_{d-1}} = \prod_{k=1}^{d-1} \phi_{i_k}(x_k^*), \qquad 0 \le i_k \le \tau_k.$$

[3]In one dimension we say that an $N \times 1$ vector v is in Vandermonde form if there is an $x \in \mathbb{C}$ such that $v_i = \phi_i(x)$ for $0 \le i \le N-1$. In higher dimensions, the vector $\operatorname{vec}(A)$ is in Vandermonde form if $A_{i_1,\dots,i_d} = \prod_{k=1}^{d} \phi_{i_k}(x_k)$ for some $x_1, \dots, x_d \in \mathbb{C}$.

Then, $\mathrm{vec}(V)$ *and* $\mathrm{vec}(W)$ *are the right and left eigenvectors of* $R_{Cayley}(p_1[x_d^*],\ldots,p_d[x_d^*])$ *corresponding to the eigenvalue* x_d^*.

Proof. Let $f_{Cayley} = f_{Cayley}(p_1[x_d^*],\ldots,p_d[x_d^*])$ be the Cayley function associated with $p_1[x_d^*],\ldots,p_d[x_d^*]$. From (3.1) we find that $f_{Cayley}(s_1,\ldots,s_{d-1},x_1^*,\ldots,x_{d-1}^*)=0$ because the determinant of a matrix with a vanishing last row is zero. Moreover, by Lemma 3.3 we have

$$0 = f_{Cayley}(s_1,\ldots,s_{d-1},x_1^*,\ldots,x_{d-1}^*) = \sum_{i_1=0}^{\tau_1}\cdots\sum_{i_{d-1}=0}^{\tau_{d-1}} Y_{i_1,\ldots,i_{d-1}}\prod_{k=1}^{d-1}\phi_{i_k}(s_k).$$

Since $\{\phi_0,\phi_1,\ldots,\}$ is a polynomial basis we must conclude that $Y=0$, and hence, $R_{Cayley}(x_d^*)v=0$ with $v=\mathrm{vec}(V)$. In other words, v is a right eigenvector of R_{Cayley} corresponding to the eigenvalue x_d^* (see Definition 2.8).

An analogous derivation shows that $\mathrm{vec}(W)$ is a left eigenvector of R_{Cayley}. □

3.3. On the generalized Rayleigh quotient of the Cayley resultant matrix. To bound $\kappa(x_d^*,R_{Cayley})$ we need to bound the absolute value of the generalized Rayleigh quotient of $R'_{Cayley}(x_d)$ (see Lemma 2.11), whenever $\underline{x}^*\in\mathbb{C}^d$ is such that x_d^* is a simple eigenvalue of $R_{Cayley}(x_d)$, i.e., there are no other solutions to (1.1) with the same dth component. In a similar style to the proof of Lemma 3.5 we show this by exploiting the relation between evaluating the derivative of f_{Cayley} and matrix-vector products with $R'_{Cayley}(x_d)$.

THEOREM 3.6. *Let* $p_1,\ldots,p_d$ *be the polynomials in* (1.1), $\underline{x}^*\in\mathbb{C}^d$ *a solution of* (1.1), *and* $f_{Cayley}(x_d)$ *the Cayley function associated with* $q_1=p_1[x_d],\ldots,q_d=p_d[x_d]$. *We have*

$$f'_{Cayley}(x_d^*)\Big|_{\substack{s_k=t_k=x_k^*\\1\le k\le d-1}} = \det(J(x_d^*)),$$

where $J(\underline{x}^*)$ *is the Jacobian matrix in* (2.1). *That is,* $f'_{Cayley}(x_d^*)$ *evaluated at* $s_k=t_k=x_k^*$ *for* $1\le k\le d-1$ *is equal to the determinant of the Jacobian.*

Proof. Recall from (3.1) that $f_{Cayley}(x_d)$ is a polynomial in $s_1,\ldots,s_{d-1},t_1,\ldots,t_{d-1}$ written in terms of a matrix determinant, and set $q_1=p_1[x_d],\ldots,q_d=p_d[x_d]$. The determinant in (3.1) for $f_{Cayley}(x_d)$ can be expanded to obtain

$$f_{Cayley}(x_d) = \frac{1}{\prod_{i=1}^{d-1}(s_i-t_i)}\sum_{\sigma\in S_d}(-1)^\sigma\prod_{i=1}^{d}p_{\sigma_i}[x_d](t_1,\ldots,t_{i-1},s_i,\ldots,s_{d-1}),$$

where S_d is the symmetric group of $\{1,\ldots,d\}$ and $(-1)^\sigma$ is the signature of the permutation σ. When we evaluate $f_{Cayley}(x_d)$ at $s_k=t_k=x_k^*$ for $1\le k\le d-1$ the denominator vanishes, and hence, so does the numerator because $f_{Cayley}(x_d)$ is a polynomial. Thus, by L'Hospital's rule, $f'_{Cayley}(x_d^*)$ evaluated $s_k=t_k=x_k^*$ for $1\le k\le d-1$ is equal to

$$\frac{\partial^d}{\partial s_1\cdots\partial s_{d-1}\partial x_d}\sum_{\sigma\in S_d}(-1)^\sigma\prod_{i=1}^{d}p_{\sigma_i}[x_d](t_1,\ldots,t_{i-1},s_i,\ldots,s_{d-1}) \tag{3.3}$$

evaluated at $s_k=x_k^*$, $t_k=x_k^*$, and $x_d=x_d^*$. In principle, one could now apply the product rule and evaluate the combinatorially many terms in (3.3). Instead, we note that after applying the product rule a term is zero if it contains $p_{\sigma_i}(\underline{x}^*)$ for any $\sigma\in S_d$

and $1 \leq i \leq d$ (since $\underline{x}^*$ is a solution to (1.1)). There are precisely d partial derivatives and d terms in each product so that any nonzero term when expanding 3.3 has each p_k differentiated precisely once. Finally, note that for each $1 \leq k \leq d-1$ only the $1 \leq i \leq k$ terms in the product depend on s_k. Hence, from (3.3) we obtain

$$f'_{Cayley}(x_d^*)\Big|_{\substack{s_k=t_k=x_k^* \\ 1\leq k\leq d-1}} = \sum_{\sigma\in S_d} (-1)^\sigma \prod_{i=1}^{d} \frac{\partial p_{\sigma_i}}{\partial x_i}(\underline{x}^*).$$

The result follows because the last expression is the determinant of the Jacobian matrix evaluated at $\underline{x}^*$. □

As a consequence of Theorem 3.6 we have the following unavoidable conclusion that mathematically explains the numerical difficulties that practitioners have been experiencing with hidden-variable resultant methods based on the Cayley resultant.

THEOREM 3.7. *Let $d \geq 2$. Then, there exist $p_1, \ldots, p_d$ in (1.1) with a simple root $\underline{x}^* \in \mathbb{C}^d$ such that*

$$\kappa(x_d^*, R_{Cayley}) \geq \|J(\underline{x}^*)^{-1}\|_2^d$$

and $\|J(\underline{x}^*)^{-1}\|_2 > 1$. Thus, an eigenvalue of $R_{Cayley}(x_d)$ can be more sensitive to perturbations than the corresponding root by a factor that grows exponentially with d.

Proof. Using Lemma 3.3, Theorem 3.6 has the following equivalent matrix form:

$$w^T R'_{Cayley}(x_d^*)v = \det(J(\underline{x}^*)),$$

where $v = \mathrm{vec}(V)$, $w = \mathrm{vec}(W)$, and V and W are given in Lemma 3.5. Since $\phi_0 = 1$, we know that $\|v\|_2 \geq 1$ and $\|w\|_2 \geq 1$. Hence, by Lemma 2.11

$$\kappa(x_d^*, R_{Cayley}) \geq |\det(J(\underline{x}^*))|^{-1}.$$

Denoting the singular values [26, Sec. 7.3] of the matrix $J(\underline{x}^*)$ by σ_i , select $p_1, \ldots, p_d$ and $\underline{x}^* \in \mathbb{C}^d$ such that $|\det(J(\underline{x}^*))| = \prod_{i=1}^d \sigma_i = \sigma_d^d$. Such polynomial systems do exist, for example, linear polynomial systems where $M\underline{x} - M\underline{x}^* = 0$ and M is a matrix with singular values $\sigma_1 = \sigma_2 = \cdots = \sigma_d$. To ensure that $\|J(\underline{x}^*)^{-1}\|_2 > 1$ we also require $\sigma_d < 1$. Then, we have

$$\kappa(x_d^*, R_{Cayley})^{-1} \leq |\det(J(\underline{x}^*))| = \prod_{i=1}^{d} \sigma_i = \sigma_d^d = \|J(\underline{x}^*)^{-1}\|_2^{-d}.$$

The result follows. □

We emphasize that this numerical instability is truly spectacular, affects the accuracy of x_d^*, and can grow exponentially with the dimension d.

Moreover, Theorem 3.7 holds for any degree-graded polynomial basis selected to represent $p_1, \ldots, p_d$ as long as $\phi_0 = 1$. In particular, the associated numerical instability cannot be resolved in general by a special choice of polynomial basis.

Theorem 3.7 is pessimistic and importantly does not imply that the resultant method always loses accuracy, just that it might. In general, one must know the solutions to (1.1) and the singular values of the Jacobian matrix to be able to predict if and when the resultant method will be accurate.

One should note that Theorem 3.7 concerns absolute conditioning and one may may wonder if a similar phenomenon also occurs in the relative sense. In Section 5 we show that the relative conditioning can also be increased by an exponential factor with d.

4. The Sylvester resultant is numerically unstable for bivariate rootfinding. A popular alternative in two dimensions to the Cayley resultant matrix is the Sylvester resultant matrix [15, Chap. 3], denoted here by R_{Sylv}. We now set out to show that the hidden-variable resultant based on R_{Sylv} is also numerically unstable. However, since $d = 2$ the instability has only a moderate impact in practice as the conditioning can only be at most squared. With care, practical bivariate rootfinders can be based on the Sylvester resultant [39] though there is the possibility that a handful digits are lost.

A neat way to define the Sylvester resultant matrix that accommodates nonmonomial polynomial bases is to define the matrix one row at a time.

DEFINITION 4.1 (Sylvester resultant matrix). *Let q_1 and q_2 be two univariate polynomials in $\mathbb{C}_n[x_1]$ of degree exactly τ_1 and τ_2, respectively. Then, the Sylvester resultant matrix $R_{Sylv} \in \mathbb{C}^{(\tau_1+\tau_2)\times(\tau_1+\tau_2)}$ associated with q_1 and q_2 is defined row-by-row as*

$$R_{Sylv}\,(i,:) = Y^{i,1}, \qquad 0 \le i \le \tau_2 - 1,$$

where $Y^{i,1}$ is the row vector of coefficients such that $q_1(x)\phi_i(x) = \sum_{k=0}^{\tau_1+\tau_2-1} Y_k^{i,1}\phi_k(x)$ and

$$R_{Sylv}\,(i+\tau_2,:) = Y^{i,2}, \quad 0 \le i \le \tau_1 - 1,$$

where $Y^{i,2}$ is the row vector of coefficients such that $q_2(x)\phi_i(x) = \sum_{k=0}^{\tau_1+\tau_2-1} Y_k^{i,2}\phi_k(x)$.

In the monomial basis, i.e., $\phi_k(x) = x^k$, Definition 4.1 gives the following Sylvester[4] matrix of size $(\tau_1 + \tau_2) \times (\tau_1 + \tau_2)$ [15, Chap. 3]:

$$R_{Sylv} = \begin{pmatrix} a_0 & a_1 & \dots & a_{\tau_1} & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & a_0 & a_1 & \dots & a_{\tau_1} \\ b_0 & b_1 & \dots & b_{\tau_2} & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & b_0 & b_1 & \dots & b_{\tau_2} \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} \tau_2 \text{ rows} \\ \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} \tau_1 \text{ rows} \end{matrix} \tag{4.1}$$

where $q_1(x) = \sum_{k=0}^{\tau_1} a_k x^k$ and $q_2(x) = \sum_{k=0}^{\tau_2} b_k x^k$.

4.1. A generalization of Clenshaw's algorithm for degree-graded polynomial bases. Our goal is to use Lemma 2.11 to bound the condition number of the eigenvalues of the Sylvester resultant matrix. It turns out the right eigenvectors of R_{Sylv} are in Vandermonde form. However, the left eigenvectors have a more peculiar structure and are related to the byproducts of a generalized Clenshaw's algorithm for degree-graded polynomial bases (see Lemma 4.4). We develop a Clenshaw's algorithm for degree-graded bases in this section with derivations of its properties in Appendix A.

[4]Variants of (4.1) include its transpose or a permutation of its rows and/or columns. Our analysis still applies after these aesthetic modifications with an appropriate change of indices. We have selected this variant for the convenience of indexing notation.

The selected polynomial basis $\phi_0, \phi_1, \ldots,$ is degree-graded and hence, satisfies a recurrence relation of the form

$$\phi_{k+1}(x) = (\alpha_k x + \beta_k)\phi_k(x) + \sum_{j=1}^{k} \gamma_{k,j}\phi_{j-1}(x), \qquad k \geq 1, \tag{4.2}$$

where $\phi_1(x) = (\alpha_0 x + \beta_0)\phi_0(x)$ and $\phi_0(x) = 1$. If $\phi_0, \phi_1, \ldots,$ is an orthogonal polynomial basis, then (4.2) is a three-term recurrence and it is standard to employ Clenshaw's algorithm [14] to evaluate polynomials expressed as $p(x) = \sum_{k=0}^{n} a_k\phi_k(x)$. This procedure can be extended to any degree-graded polynomial basis.

Let $p(x)$ be expressed as $p(x) = \sum_{k=0}^{n} a_k\phi_k(x)$, where $\phi_0, \ldots, \phi_n$ is a degree-graded polynomial basis. One can evaluate $p(x)$ via the following procedure: Let $b_{n+1}[p](x) = 0$, and calculate $b_n[p](x), \ldots, b_1[p](x)$ from the following recurrence relation:

$$b_k[p](x) = a_k + (\alpha_k x + \beta_k)b_{k+1}[p](x) + \sum_{j=k+1}^{n-1} \gamma_{j,k+1}b_{j+1}[p](x), \qquad 1 \leq k \leq n. \tag{4.3}$$

We refer to the quantities $b_1[p](x), \ldots, b_{n+1}[p](x)$ as *Clenshaw shifts* (in the monomial case they are called Horner shifts [16]). The value $p(x)$ can be written in terms of the Clenshaw shifts[5].

LEMMA 4.2. *Let n be an integer, $x \in \mathbb{C}$, $\phi_0, \ldots, \phi_n$ a degree-graded basis satisfying* (4.2)*, $p(x) = \sum_{k=0}^{n} a_k\phi_k(x)$, and $b_{n+1}[p](x), \ldots, b_1[p](x)$ the Clenshaw shifts satisfying* (4.3)*. Then,*

$$p(x) = a_0\phi_0(x) + \phi_1(x)b_1[p](x) + \sum_{i=1}^{n-1} \gamma_{i,1}b_{i+1}[p](x). \tag{4.4}$$

Proof. See Appendix A. □

Clenshaw's algorithm for degree-graded polynomial bases is summarized in Figure 2. We note that because of the full recurrence in (4.3) the algorithm requires $\mathcal{O}(n^2)$ operations to evaluate $p(x)$. Though this algorithm may not be of significant practical importance, it is of theoretical interest for the conditioning analysis of some linearizations from the so-called $\mathbb{L}_1$- or $\mathbb{L}_2$-spaces [31] when degree-graded bases are employed [34].

There is a remarkable and interesting connection between Clenshaw shifts and the quotient $(p(x)-p(y))/(x-y)$, which will be useful when deriving the left eigenvectors of R_{Sylv}.

THEOREM 4.3. *With the same set up as Lemma 4.2 we have*

$$\frac{p(x) - p(y)}{x - y} = \sum_{i=0}^{n-1} \alpha_i b_{i+1}[p](y)\phi_i(x), \quad x \neq y \tag{4.5}$$

and

$$p'(x) = \sum_{i=0}^{n-1} \alpha_i b_{i+1}[p](x)\phi_i(x). \tag{4.6}$$

[5]Note that, although Lemma 4.2 is stated in a general form and holds for *any* degree-graded basis, in this paper we fix the normalization $\max_{x\in\Omega} |\phi_j(x)| = 1$, that implies in particular $\phi_0 = 1$ simplifying (4.2).

Clenshaw's algorithm for degree-graded polynomial bases

Let $\phi_0, \phi_1, \ldots,$ satisfy (4.2) and $p(x) = \sum_{k=0}^{n} a_k \phi_k(x)$.

Set $b_{n+1}[p](x) = 0$.

for $k = n, n-1, \ldots, 1$ **do**

$b_k[p](x) = a_k + (\alpha_k x + \beta_k) b_{k+1}[p](x) + \sum_{j=k+1}^{n-1} \gamma_{j,k+1} b_{j+1}[p](x)$

end

$p(x) = a_0\phi_0(x) + \phi_1(x) b_1[p](x) + \sum_{j=1}^{n-1} \gamma_{j,1} b_{j+1}[p](x)$.

FIG. 2. *Clenshaw's algorithm for evaluating polynomials expressed in a degree-graded basis.*

Proof. See Appendix A. □

The relation between the derivative and Clenshaw shifts in (4.6) has been noted by Skrzipek for orthogonal polynomial bases in [37], where it was used to construct a so-called *extended Clenshaw's algorithm* for evaluating polynomial derivatives. Using Theorem 4.3 and [37] an extended Clenshaw's algorithm for polynomials expressed in a degree-graded basis is immediate.

4.2. The eigenvector structure of the Sylvester resultant matrix. We now set $q_1 = p_1[x_2]$ and $q_2 = p_2[x_2]$ (considering x_2 as the hidden variable), and we are interested in the eigenvectors of the matrix polynomial $R_{Sylv}(x_2^*)$, when (x_1^*, x_2^*) is a solution to (1.1) when $d = 2$. It turns out that the right eigenvectors of $R_{Sylv}(x_2^*)$ are in Vandermonde form, while the left eigenvectors are related to the Clenshaw shifts (see Section 4.1).

LEMMA 4.4. *Suppose that $\underline{x}^* = (x_1^*, x_2^*)$ is a simple root of (1.1) and that $p_1[x_2]$ and $p_2[x_2]$ are of degree τ_1 and τ_2, respectively, in x_1. The right eigenvector of $R_{Sylv}(x_2^*)$ corresponding to the eigenvalue x_2^* is*

$$v_k = \phi_k(x_1^*), \qquad 0 \le k \le \tau_1 + \tau_2 - 1,$$

and the left eigenvector is defined as

$$w_i = \begin{cases} -\alpha_i b_{i+1}[q_2](x_1^*), & 0 \le i \le \tau_2 - 1, \\ \alpha_{i-\tau_2} b_{i-\tau_2+1}[q_1](x_1^*), & \tau_2 \le i \le \tau_1 + \tau_2 - 1, \end{cases}$$

where $q_j = p_j[x_2^*]$ and $b_k[q_j](x_1^*)$ are the Clenshaw shifts with respect to $\{\phi_0, \phi_1, \ldots, \}$.

Proof. By construction we have, for $0 \le i \le \tau_2 - 1$,

$$R_{Sylv}(i, :)\, v = \sum_{k=0}^{\tau_1+\tau_2-1} Y_k^{i,1}(x_2^*)\phi_k(x_1^*) = q_1(x_1^*)\phi_i(x_1^*) = 0$$

and, for $0 \le i \le \tau_1 - 1$,

$$R_{Sylv}(i + \tau_2, :)\, v = \sum_{k=0}^{\tau_1+\tau_2-1} Y_k^{i,2}(x_2^*)\phi_k(x_1^*) = q_2(x_1^*)\phi_i(x_1^*) = 0.$$

Thus, v is a right eigenvector of $R_{Sylv}(x_2^*)$ corresponding to the eigenvalue x_2^*.

For the left eigenvector, first note that for any vector Φ of the form $\Phi_k = \phi_k(x)$ for $0 \le k \le \tau_1 + \tau_2 - 1$ we have by Theorem 4.3

$$\begin{aligned} w^T R_{Sylv}(x_2^*)\Phi &= -\sum_{i=0}^{\tau_2-1} \alpha_i b_{i+1}[q_2](x_1^*)\phi_i(x)q_1(x) + \sum_{i=0}^{\tau_1-1} \alpha_i b_{i+1}[q_1](x_1^*)\phi_i(x)q_2(x) \\ &= -\frac{q_2(x) - q_2(x_1^*)}{x - x_1^*} q_1(x) + \frac{q_1(x) - q_1(x_1^*)}{x - x_1^*} q_2(x) \\ &= -\frac{q_2(x)}{x - x_1^*} q_1(x) + \frac{q_1(x)}{x - x_1^*} q_2(x) = 0, \end{aligned}$$

where the second from last equality follows because $q_1(x_1^*) = q_2(x_1^*) = 0$. Since (4.3) holds for any x and $\{\phi_0, \phi_1, \ldots, \phi_{\tau_1+\tau_2-1}\}$ is a basis of $\mathbb{C}_{\tau_1+\tau_2-1}[x]$, we deduce that $w^T R_{Sylv}(x_2^*) = 0$, and hence, w is a left eigenvector of R_{Sylv} corresponding to the eigenvalue x_2^*. □

4.3. On the generalized Rayleigh quotient of the Sylvester resultant matrix. To bound $\kappa(R_{Sylv}, x_d^*)$ we look at the absolute value of the generalized Rayleigh quotient of $R'_{Sylv}(x_2^*)$, whenever $\underline{x}^*$ is such that x_2^* is a simple eigenvalue of $R_{Sylv}(x_2)$. Lemma 4.4 allows us to show how the generalized Rayleigh quotient of $R'_{Sylv}(x_2^*)$ relates to the determinant of the Jacobian.

LEMMA 4.5. *With the same assumptions as in Lemma 4.4, we have*

$$\frac{|w^T R'_{Sylv}(x_2^*)v|}{\|v\|_2\|w\|_2} \le \frac{|\det(J(\underline{x}^*))|}{\|w\|_2},$$

where w and v are the left and right eigenvectors of R_{Sylv}, respectively, and $J(\underline{x}^*)$ is the Jacobian matrix in (2.1).

Proof. By Lemma 4.4 we know the structure of v and w. Hence, we have

$$\begin{aligned} w^T R'_{Sylv}(x_2^*)v &= -\sum_{i=0}^{\tau_2-1} \alpha_i b_{i+1}[q_2](x_1^*)\phi_i(x_1^*)\frac{\partial q_1}{\partial x_2}(x_1^*) + \sum_{i=0}^{\tau_1-1} \alpha_i b_{i+1}[q_1](x_1^*)\phi_i(x_1^*)\frac{\partial q_2}{\partial x_2}(x_1^*) \\ &= -\frac{\partial q_1}{\partial x_2}(x_1^*)\frac{\partial q_2}{\partial x_1}(x_1^*) + \frac{\partial q_1}{\partial x_1}(x_1^*)\frac{\partial q_2}{\partial x_2}(x_1^*), \end{aligned}$$

where the last equality used the relation in (4.6). The result now follows since this final expression equals $\det(J(\underline{x}^*))$ and since $\phi_0 = 1$ we have $\|v\|_2 \ge 1$. □

THEOREM 4.6. *There exist p_1 and p_2 in* (1.1) *with a simple root $\underline{x}^* \in \mathbb{C}^2$ such that*

$$\kappa(x_2^*, R_{Sylv}) \ge \|J(\underline{x}^*)^{-1}\|_2^2$$

and $\|J(\underline{x}^*)^{-1}\|_2 > 1$. Thus, an eigenvalue of $R_{Sylv}(x_2)$ can be squared more sensitive to perturbations than the corresponding root in the absolute sense.

Proof. We give an example for which $\|w\|_2 \ge 1$ in Lemma 4.5. For some positive parameter u and for some $n \ge 2$ consider the polynomials

$$p_1(x_1, x_2) = x_1^n x_2^n + u^{1/2} x_1, \quad p_2(x_1, x_2) = \alpha_{n-1}^{-1}(x_1^n + x_2^n) + u^{1/2} x_2.$$

It is trivial to verify that $\underline{x}^* = (0, 0)$ is a common root[6]. Since $|b_n[q_2](0)| = \alpha_{n-1}\alpha_{n-1}^{-1} = 1$ we have $\|w\|_2 \ge 1$. The result then follows from $|\det(J(\underline{x}^*))| = \|J(\underline{x}^*)^{-1}\|_2^{-2}$ and Lemma 4.5. □

[6] By a change of variables, there is an analogous example with a solution anywhere in the complex plane.

Theorem 4.6 mathematically explains the numerical difficulties that practitioners have been experiencing with hidden-variable resultant methods based on the Sylvester resultant. There are successful bivariate rootfinders based on this methodology [39] for low degree polynomial systems and it is a testimony to those authors that they have developed algorithmic remedies (not cures) for the inherent numerical instability.

We emphasize that Theorem 4.6 holds for any normalized degree-graded polynomial basis. Thus, the mild numerical instability cannot, in general, be overcome by working in a different degree-graded polynomial basis.

The example in the proof of Theorem 4.6 is quite alarming for a practioner since if u is the unit machine roundoff, then we have $\|J(0,0)^{-1}\|_2 = u^{-1/2}$ and $\kappa(x_2^*, R_{Sylv}) = u^{-1}$. Thus, a numerical rootfinder based on the Sylvester resultant matrix may entirely miss a solution that has a condition number larger than $u^{-1/2}$. A stable rootfinder should not miss such a solution.

When $d = 2$, we can use Theorem 3.7 and Lemma 4.5 to conclude that the ratio between the conditioning of the Cayley and Sylvester resultant matrices for the same eigenvalue x_2^* is equal to $\|v\|_2/\|w\|_2$, where v and w are the right and left eigenvector of $R_{Sylv}(x_2^*)$ associated with the eigenvalue x_2^*. This provides theoretical support for the numerical observations in [35]. However, it seems difficult to predict *a priori* if the Cayley or Sylvester resultant matrix will behave better numerically. For real polynomials and $d = 2$, the Cayley resultant matrix is symmetric and this structure can be exploited [35]. In the monomial basis, the Sylvester resultant matrix is two stacked Toeplitz matrices (see (4.1)). It may be that structural differences like these are more important than their relatively similar numerical properties when $d = 2$.

5. A discussion on relative and absolute conditioning. Let $X(D)$ be the solution of a mathematical problem depending on data D. In general, with the very mild assumption that D and X lie in Banach spaces, it is possible to define the absolute condition number of the problem by perturbing the data to $D + \delta D$ and studying the behaviour of the perturbed solution $\hat{X}(D + \delta D) = X(D) + \delta X(D, \delta D)$:

$$\kappa_{\text{abs}} = \lim_{\epsilon \to 0} \sup_{\|\delta D\| \leq \epsilon} \frac{\|\delta X\|}{\|\delta D\|}.$$

Similarly, a relative condition number can be defined by looking at the limit ratios of *relative* changes.

$$\kappa_{\text{rel}} = \lim_{\epsilon \to 0} \sup_{\|\delta D\| \leq \epsilon \|D\|} \frac{\|\delta X\|}{\|\delta D\|} \frac{\|D\|}{\|X\|} = \kappa_{\text{abs}} \frac{\|D\|}{\|X\|}.$$

In this paper, we have compared two absolute condition numbers. One is given by Proposition 2.9: there, $X = \underline{x}^*$ is a solution of (1.1) while $D = (p_1, \ldots, p_d)$ is the set of polynomials in (1.1). The other is given by Lemma 2.11, where D is a matrix polynomial and $X = x_d^*$ is the dth component of $\underline{x}^*$.

To quote N. J. Higham [25, p. 56]: "Usually, it is the relative condition number that is of interest, but it is more convenient to state results for the absolute condition number". This remark applies to our analysis as well. We have found it convenient to study the absolute condition number, but when attempting to solve the rootfinding problem in floating point arithmetic it is natural to allow for relatively small perturbations, and thus to study the relative condition number. Hence, a natural question is whether the exponential increase of the absolute condition number in Theorem 3.7 and the squaring in Theorem 4.6 causes a similar effect in the relative condition number.

It is not immediate that the exponential increase of the absolute condition number leads to the same effect in the relative sense. We have found examples where the exponential increase of the absolute condition number is perfectly counterbalanced by an exponentially small Cayley resultant matrix. For instance, linear polynomial systems, when the Cayley resultant method is equivalent to Cramer's rule, seems to fall into this category. In the relative sense, it may be possible to show that the hidden-variable resultant method based on Cayley or Sylvester is either numerically unstable during the construction of the resultant matrix or the resultant matrix has an eigenvalue that is more sensitive to small relative perturbations than hoped. We do not know yet how to make such a statement precise.

Instead, we provide an example that shows that the hidden-variable resultant method remains numerically unstable in the relative sense. Let u be a sufficiently small real positive parameter and $d \geq 2$. Consider the following polynomial system:

$$\begin{aligned} p_{2i-1}(\underline{x}) &= x_{2i-1}^2 + u\left(\tfrac{\sqrt{2}}{2}x_{2i-1} + \tfrac{\sqrt{2}}{2}x_{2i}\right), \\ p_{2i}(\underline{x}) &= x_{2i}^2 + u\left(\tfrac{\sqrt{2}}{2}x_{2i} - \tfrac{\sqrt{2}}{2}x_{2i-1}\right), \qquad 1 \leq i \leq \lfloor d/2 \rfloor, \end{aligned}$$

where if d is odd then take $p_d(\underline{x}) = x_d^2 + ux_d$. Selecting $\Omega = [-1,1]^d$, we have that $\|p_i\|_\infty = 1 + \sqrt{2}u$ for $1 \leq i \leq d$, except possibly $\|p_d\|_\infty = 1 + u$ if d is odd. It can be shown that the origin[7], $\underline{x}^*$, is a simple root, $\det(J(\underline{x}^*)) = u^d$, $\|J(\underline{x}^*)^{-1}\|_2 = u^{-1}$, and that

$$f_{Cayley}(s_1, \ldots, s_{d-1}, t_1, \ldots, t_{d-1}) = \prod_{k=1}^{d-1}(s_k + t_k)x_d^2 + \mathcal{O}(u).$$

Thus, neither the polynomials p_i or the resultant matrix $R_{Cayley}(x_d)$ is small. In such an example, the relative condition number will exhibit the same behavior as the absolute condition number. In particular, the relative condition number of an eigenvalue of $R_{Cayley}(x_d)$ may be larger than the relative condition number of the corresponding solution by a factor that grows exponentially with d.

The same example (for $d = 2$), and a similar argument, applies to the Sylvester resultant matrix showing the conditioning can be squared in the relative sense too.

6. Future outlook. In this paper we have shown that two popular hidden-variable resultant methods are numerically unstable. We suspect that many more variants of hidden-variable resultant methods are numerically unstable. In particular, we hesitantly suggest that it could be the case that almost all practical resultants have numerical issues. We do not know exactly how to formulate such a general statement, but we note that practitioners are widely experiencing problems with hidden-variable resultant methods. In particular, we do not know of a numerical multidimensional rootfinder based on resultants that is robust for polynomial systems of large degree and high d.

We would like to find a resultant matrix that can be constructed numerically and that provably does not lead to a numerically unstable hidden-variable resultant method. We do not know of one. In principle, this could be a breakthrough in global rootfinding with significant practical applications as it might allow (1.1) to be converted into a large eigenproblem without confronting conditioning issues. Solving

[7] By a change of variables, there is an analogous example with a solution anywhere in $[-1,1]^d$.

high-dimensional and large degree polynomial systems would then be restricted by computational cost rather than numerical accuracy.

Finally, we express again our hope that this paper, while appearing rather negative, will have a positive impact on future research into numerical rootfinders. We believe that with care resultant-based methods can become a practical tool for multidimensional rootfinding.

Acknowledgments. We thank Yuji Nakatsukasa, one of our closest colleagues, for his insightful discussions during the writing of [35] that ultimately lead us to consider conditioning issues more closely. We also thank Anthony Austin and Martin Lotz for carefully reading a draft and providing us with excellent comments. While this manuscript was in a much earlier form Martin Lotz personally sent it to Gregorio Malajovich for his comments. Gregorio's comprehensive and enthusiastic reply encouraged us to proceed with renewed vigor.

Appendix A. A generalization of Clenshaw's algorithm for degree-graded polynomial bases. This appendix contains the tedious, though necessary, proofs required in Section 4.1 for Clenshaw's algorithm for evaluating polynomials expressed in a degree-graded basis.

Proof. [Proof of Lemma 4.2] By rearranging (4.3) we have $a_k = b_k[p](x) - (\alpha_k x + \beta_k)b_{k+1}[p](x) - \sum_{j=k+1}^{n-1} \gamma_{j,k+1} b_{j+1}[p](x)$. Thus,

$$p(x) = a_0\phi_0(x) + \sum_{k=1}^{n} \left[b_k[p](x) - (\alpha_k x + \beta_k)b_{k+1}[p](x) - \sum_{j=k+1}^{n-1} \gamma_{j,k+1} b_{j+1}[p](x) \right] \phi_k(x).$$

Now, by interchanging the summations and collecting terms we have

$$\begin{aligned} p(x) &= a_0\phi_0(x) + \sum_{k=1}^{n} \phi_k(x) b_k[p](x) - \sum_{k=2}^{n} (\alpha_{k-1}x + \beta_{k-1})\, \phi_{k-1}(x) b_k[p](x) \\ &\qquad - \sum_{j=2}^{n-1} \left[\sum_{k=1}^{j-1} \gamma_{j,k+1}\phi_k(x) \right] b_{j+1}[p](x) \\ &= a_0\phi_0(x) + \phi_1(x)b_1[p](x) \\ &\qquad + \sum_{j=1}^{n-1} \left[\phi_{j+1}(x) - (\alpha_j x + \beta_j)\phi_j(x) - \sum_{k=1}^{j-1} \gamma_{j,k+1}\phi_k(x) \right] b_{j+1}[p](x) \end{aligned}$$

Finally, using (4.2) we obtain

$$p(x) = a_0\phi_0(x) + \phi_1(x)b_1[p](x) + \sum_{j=1}^{n-1} \gamma_{j,1}\phi_0(x) b_{j+1}[p](x),$$

as required. ∎

Section 4.1 also shows that Clenshaw's algorithm connects to the the quotient $(p(x) - p(y))/(x - y)$. To achieve this we need an immediate result that proves a different recurrence relation on the Clenshaw shifts to (4.3). The proof involves tedious algebraic manipulations and mathematical strong induction.

LEMMA A.1. *Let n be an integer, $\phi_0, \ldots, \phi_n$ a degree-graded basis satisfying* (4.2)*, and $b_{n+1}[p], \ldots, b_1[p]$ the Clenshaw shifts satisfying* (4.3)*. Then, for $1 \le j \le n$,*

$$b_j[\phi_{n+1}](x) = (\alpha_n x + \beta_n) b_j[\phi_n](x) + \sum_{s=j+1}^{n} \gamma_{n,s} b_j[\phi_{s-1}](x).$$

Proof. We proceed by induction on j. Let $j = n$. We have, by (4.3),

$$b_n[\phi_{n+1}](x) = (\alpha_n x + \beta_n) b_{n+1}[\phi_{n+1}](x) = (\alpha_n x + \beta_n) b_n[\phi_n](x),$$

where the last equality follows because $b_{n+1}[\phi_{n+1}](x) = b_n[\phi_n](x) = 1$ Now, suppose the result holds for $j = n, n-1, \ldots, k+1$. We have, by (4.3) and the inductive hypothesis,

$$\begin{aligned}
b_k[\phi_{n+1}](x) &= (\alpha_k x + \beta_k) b_{k+1}[\phi_{n+1}](x) + \sum_{j=k+1}^{n} \gamma_{j,k+1} b_{j+1}[\phi_{n+1}](x) \\
&= (\alpha_k x + \beta_k) \left[(\alpha_n x + \beta_n) b_{k+1}[\phi_n](x) + \sum_{s=k+2}^{n} \gamma_{n,s} b_{k+1}[\phi_{s-1}](x) \right] \\
&\quad + \sum_{j=k+1}^{n-1} \gamma_{j,k+1} \left[(\alpha_n x + \beta_n) b_{j+1}[\phi_n](x) + \sum_{s=j+2}^{n} \gamma_{n,s} b_{j+1}[\phi_{s-1}](x) \right] \\
&\quad + \gamma_{n,k+1} b_{n+1}[\phi_{n+1}](x).
\end{aligned}$$

By interchanging the summations and collecting terms we have

$$\begin{aligned}
b_k[\phi_{n+1}](x) &= (\alpha_n x + \beta_n) \left[(\alpha_k x + \beta_k) b_{k+1}[\phi_n](x) + \sum_{j=k+1}^{n-1} \gamma_{j,k+1} b_{j+1}[\phi_n](x) \right] \\
&\quad + \sum_{s=k+3}^{n} \gamma_{n,s} \left[(\alpha_k x + \beta_k) b_{k+1}[\phi_{s-1}](x) + \sum_{j=k+1}^{s-2} \gamma_{j,k+1} b_{j+1}[\phi_{s-1}](x) \right] \\
&\quad + \gamma_{n,k+2} (\alpha_k x + \beta_k) b_{k+1}[\phi_{k+1}](x) + \gamma_{n,k+1} b_{n+1}[\phi_{n+1}](x) \\
&= (\alpha_n x + \beta_n) b_k[\phi_n] + \sum_{s=k+1}^{n} \gamma_{n,s} b_k[\phi_{s-1}],
\end{aligned}$$

where in the last equality we used (4.3), $(\alpha_k x + \beta_k) b_{k+1}[\phi_{k+1}](x) = b_k[\phi_{k+1}](x)$, and $b_{n+1}[\phi_{n+1}](x) = b_k[\phi_k](x) = 1$. □

The recurrence from Lemma A.1 allows us to prove Theorem 4.3.

Proof. [Proof of Theorem 4.3]

Case 1: $\mathbf{x \neq y}$. Since for a fixed y the Clenshaw shifts are linear, i.e., $b_j[c_1\phi_i + c_2\phi_k](y) = c_1 b_j[\phi_i](y) + c_2 b_j[\phi_k](y)$ for constants c_1 and c_2, it is sufficient to prove the theorem for $p = \phi_n$ for $n \geq 1$.

We proceed by induction on n. For $n = 1$ we have

$$\sum_{j=0}^{n-1} \alpha_j b_{j+1}[\phi_{n+1}](y) \phi_j = \alpha_0 b_1[\phi_1](y) = \alpha_0 = \frac{\phi_1(x) - \phi_1(y)}{x - y}.$$

Assume that the result holds for $n = 1, \ldots, k-1$. From the inductive hypothesis,

we have

$$\frac{\phi_{k+1}(x)-\phi_{k+1}(y)}{x-y} = \alpha_k\phi_k(x) + (\alpha_k x+\beta_k)\frac{\phi_k(x)-\phi_k(y)}{x-y} + \sum_{j=1}^{k}\gamma_{k,j}\frac{\phi_{j-1}(x)-\phi_{j-1}(y)}{x-y}$$
$$= \alpha_k\phi_k(x) + (\alpha_k x+\beta_k)\sum_{j=0}^{k-1}\alpha_j b_{j+1}[\phi_k](y)\phi_j(x)$$
$$+ \sum_{j=1}^{k}\gamma_{k,j}\sum_{s=0}^{j-2}\alpha_s b_{s+1}[\phi_{j-1}](y)\phi_s(x).$$

Moreover, by interchanging the summations and collecting terms we have

$$\frac{\phi_{k+1}(x)-\phi_{k+1}(y)}{x-y} = \alpha_k\phi_k(x) + (\alpha_k x+\beta_k)\alpha_{k-1}b_k[\phi_k](y)\phi_{k-1}(x)$$
$$+ \sum_{j=0}^{k-2}\alpha_j\left[(\alpha_k x+\beta_k)b_{j+1}[\phi_k](y) + \sum_{s=j+2}^{k}\gamma_{k,s}b_{j+1}[\phi_{s-1}](y)\right]\phi_j(x).$$

Finally, since $b_{k+1}[\phi_{k+1}](y)=1$, $b_k[\phi_{k+1}](y) = (\alpha_k x+\beta_k)b_k[\phi_k](y)$, and by (4.3), we have

$$\frac{\phi_{k+1}(x)-\phi_{k+1}(y)}{x-y} = \alpha_k b_{k+1}[\phi_{k+1}](y)\phi_k(x) + \alpha_{k-1}b_k[\phi_{k+1}](y)\phi_{k-1}(x)$$
$$+ \sum_{j=0}^{k-2}\alpha_j b_{j+1}[\phi_{k+1}](y)\phi_j(x)$$

and the result follows by induction.

Case 2: $\mathbf{x=y}$. Immediately follows from $x\neq y$ by using L'Hospital's rule on (4.5). □

REFERENCES

[1] E. L. Allgower, K. Georg, and R. Miranda, *The method of resultants for computing real solutions of polynomial systems*, SIAM J. Numer. Anal., 29 (1992), pp. 831–844.

[2] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura, *A numerical method for polynomial eigenvalue problems using contour integral*, Japan J. Indust. Appl. Math., 27 (2010), pp. 73–90.

[3] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler, *Numerically Solving Polynomial Systems with Bertini*, SIAM, 2013.

[4] D. A. Bini and A. Marco, *Computing curve intersection by means of simultaneous iterations*, Numer. Algor., 43 (2006), pp. 151–175.

[5] D. A. Bini and V. Noferini, *Solving polynomial eigenvalue problems by means of the Ehrlich-Aberth method*, Linear Algebra Appl., 439 (2013), pp. 1130–1149.

[6] D. Bondyfalat, B. Mourrain, and V. Y. Pan, *Solution of a polynomial system of equations via the eigenvector computation*, Linear Algebra Appl., 319 (2000), pp. 193–209.

[7] J. P. Boyd, *Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding*, SIAM J. Numer. Anal., 40 (2002), pp. 1666–1682.

[8] J. P. Boyd, *Solving Transcendental Equations: The Chebyshev Polynomial Proxy and Other Numerical Rootfinders*, SIAM, 2014.

[9] B. Buchberger, *An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal*, J. Symbolic Comput., 41 (2006), pp. 475–511.

[10] L. Busé, H. Khalil, and B. Mourrain, *Resultant-based methods for plane curves intersection problems*, Computer algebra in scientific computing, Springer Berlin Heidelberg, 3718 (2005), pp. 75–92.

[11] E. Cattani and A. Dickenstein, *Introduction to residues and resultants*, in, A. Dickenstein and I. Z. Emiris, editors, *Solving Polynomial Equations. Foundations, Algorithms, and Applications*, Springer, 2005.
[12] A. Cayley, *On the theory of elimination*, Cambridge and Dublin Math. J. III, (1848), pp. 116–120.
[13] E.-W. Chionh, M. Zhang, and R. N. Goldman, *Fast computation of the Bézout and Dixon resultant matrices*, J. Symb. Comput., 33 (2002), pp. 13–29.
[14] C. W. Clenshaw, *A note on the summation of Chebyshev series*, Math. Comput., 9 (1955), pp. 118–120.
[15] D. A. Cox, J. Little, and D. O'Shea, *Using Algebraic Geometry*, Springer, 2013.
[16] F. De Teran, F. M. Dopico, and D. S. Mackey, *Fiedler companion linearizations and the recovery of minimal indices*, SIAM J. Mat. Anal. Appl., 31 (2010), pp. 2181–2204.
[17] P. Dreesen, K. Batselier, and B. De Moor, *Back to the roots: Polynomial system solving, linear algebra, systems theory*, Proc. 16th IFAC Symposium on System Identification (SYSID), 2012, pp. 1203–1208.
[18] I. Z. Emiris, *Sparse elimination and applications in kinematics*, Dissertation, University of California, Berkeley, 1994.
[19] I. Z. Emiris and V. Z. Pan, *Symbolic and numeric methods for exploiting structure in constructing resultant matrices*, J. Symbolic Comput., 33 (2002), pp. 393–413.
[20] I. M. Gelfand, M. Kapranov, and A. Zelevinsky, *Discriminants, Resultants, and Multidimensional Determinants*, Springer, Birkhäuser, Boston, 2008.
[21] L. Gemignani, and V. Noferini, *The Ehrlich–Aberth method for palindromic matrix polynomials represented in the Dickson basis*, Linear Algebra Appl., 438 (2013), pp. 1645–1666.
[22] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, SIAM, Philadelphia, USA, 2009, (unabridged republication of book first published by Academic Press in 1982).
[23] I. J. Good, *The colleague matrix, a Chebyshev analogue of the companion matrix*, The Quarterly Journal of Mathematics, 12 (1961), pp. 61–68.
[24] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2nd edition, 2002.
[25] N. J. Higham, *Function of Matrices: Theory and Computation*, SIAM, 2008.
[26] R. A. Horn, and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 2nd edition, New York, 2013.
[27] G. Jónsson and S. Vavasis, *Accurate solution of polynomial equations using Macaulay resultant matrices*, Math. Comput., 74 (2005), pp. 221–262.
[28] D. Kapur and T. Saxena, *Comparison of various multivariate resultant formulations*, ACM Proceedings of the 1995 international symposium on Symbolic and algebraic computation, 1995.
[29] F. C. Kirwan, *Complex Algebraic Curves*, Cambridge University Press, Cambridge, 1992.
[30] H. Li, *A simple solution to the six-point two-view focal-length problem*, Computer VisionECCV, Springer Berlin Heidelberg, (2006), pp. 200–213.
[31] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Vector spaces of linearizations for matrix polynomials*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 971–1004.
[32] D. Manocha and J. F. Canny, *Multipolynomial resultants and linear algebra*, Papers from the international symposium on Symbolic and algebraic computation. ACM, 1992.
[33] D. Manocha and J. Demmel, *Algorithms for intersecting parametric and algebraic curves I: simple intersections*, ACM Trans. Graphics, 13 (1994), pp. 73–100.
[34] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Vector spaces of linearizations for matrix polynomials: a bivariate polynomial approach*, submitted.
[35] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Computing the common zeros of two bivariate functions via Bézout resultants*, Numer. Math., 129 (2015), pp. 181–209.
[36] S. Ragnarsson and C. F. Van Loan, *Block tensor unfoldings*, SIAM J. Mat. Anal. Appl., 33 (2012), pp. 149–169.
[37] M.-R. Skrzipek, *Polynomial evaluation and associated polynomials*, Numer. Math., 79 (1998), pp. 601–613.
[38] A. J. Sommese and C. W. Wampler, *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, 2005.
[39] L. Sorber, M. Van Barel, and L. De Lathauwer, *Numerical solution of bivariate and polyanalytic polynomial systems*, SIAM J. Numer. Anal., 52 (2014), pp. 1551–1572.
[40] M. I. Syam, *Finding all real zeros of polynomial systems using multi-resultant*, J. Comput. Appl. Math., 167 (2004), pp. 417–428.
[41] L. Taslaman, *An algorithm for quadratic eigenproblems with low rank damping* , SIAM J. Matrix Anal. Appl., 36 (2015), pp. 251–272.
[42] F. Tisseur, *Backward error and condition of polynomial eigenvalue problems*, Linear Algebra

Appl., 309 (2000), pp. 339–361.
[43] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), pp. 235–286.
[44] A. Townsend, *Computing with functions of two variables*, DPhil Thesis, The University of Oxford, 2014.
[45] A. Townsend and L. N. Trefethen, *An extension of Chebfun to two dimensions*, SIAM J. Sci. Comput., 35 (2013), C495–C518.
[46] J. Weiss, *Resultant methods for the inverse kinematics problem*, Computational kinematics, Springer Netherlands, 28 (1993), pp. 41–52.