SIAM J. SCI. COMPUT.
Vol. 38, No. 2, pp. A765–A788

ROOTS OF BIVARIATE POLYNOMIAL SYSTEMS VIA DETERMINANTAL REPRESENTATIONS[*]

BOR PLESTENJAK[†] AND MICHIEL E. HOCHSTENBACH[‡]

Abstract. We give two determinantal representations for a bivariate polynomial. They may be used to compute the zeros of a system of two of these polynomials via the eigenvalues of a two-parameter eigenvalue problem. The first determinantal representation is suitable for polynomials with scalar or matrix coefficients and consists of matrices with asymptotic order $n^2/4$, where n is the degree of the polynomial. The second representation is useful for scalar polynomials and has asymptotic order $n^2/6$. The resulting method to compute the roots of a system of two bivariate polynomials is very competitive with some existing methods for polynomials up to degree 10, as well as for polynomials with a small number of terms.

Key words. system of bivariate polynomial equations, determinantal representation, two-parameter eigenvalue problem, polynomial two-parameter eigenvalue problem

AMS subject classifications. 65F15, 65H04, 65F50, 13P15

DOI. 10.1137/140983847

1. Introduction.

In this paper, we make progress on a problem that has essentially been open since Dixon's 1902 paper [10]. It is well known that for each monic polynomial $p(x) = p_0 + p_1x + \cdots + p_{n-1}x^{n-1} + x^n$ one can construct a matrix $A \in \mathbb{C}^{n\times n}$ such that $\det(xI - A) = p(x)$. One of the options is a companion matrix (see, e.g., [21, p. 146])

$$A_p = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & & & 1 \\ -p_0 & -p_1 & \cdots & \cdots & -p_{n-1} \end{bmatrix}.$$

Thus, we can numerically compute the zeros of the polynomial p as eigenvalues of the corresponding companion matrix A_p using tools from numerical linear algebra. This approach is used in many numerical packages, for instance, in the `roots` command in MATLAB [30].

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section August 26, 2014; accepted for publication (in revised form) December 28, 2015; published electronically March 15, 2016.
http://www.siam.org/journals/sisc/38-2/98384.html
[†]Department of Mathematics, University of Ljubljana, Jadranska 19, SI-1000 Ljubljana, Slovenia (bor.plestenjak@fmf.uni-lj.si). The research was performed while this author was visiting the CASA group at TU Eindhoven.
[‡]Department of Mathematics and Computer Science, TU Eindhoven, PO Box 513, 5600 MB, The Netherlands (http://www.win.tue.nl/~hochsten/). This author's work was supported by an NWO Vidi grant.

The aim of this paper is to find a similar elegant tool for finding the zeros of a system of two bivariate polynomials of degree n,

$$\begin{aligned} p(x,y) &:= \sum_{i=0}^{n}\sum_{j=0}^{n-j} p_{ij}\, x^i\, y^j \,=0,\\ q(x,y) &:= \sum_{i=0}^{n}\sum_{j=0}^{n-j} q_{ij}\, x^i\, y^j \,=0. \end{aligned} \tag{1.1}$$

An approach analogous to the univariate case would be to construct matrices A_1, B_1, C_1, A_2, B_2, and C_2 of size $n\times n$ such that

$$\begin{aligned} \det(A_1+xB_1+yC_1) &= p(x,y),\\ \det(A_2+xB_2+yC_2) &= q(x,y). \end{aligned} \tag{1.2}$$

This would give an equivalent two-parameter eigenvalue problem [1]

$$\begin{aligned} (A_1+xB_1+yC_1)\,u_1 &= 0,\\ (A_2+xB_2+yC_2)\,u_2 &= 0 \end{aligned} \tag{1.3}$$

that could be solved by the standard tools like the QZ algorithm; see [17] for details.

This idea looks promising, but there are many obstacles on the way to a working numerical algorithm that could be applied to a system of bivariate polynomials. Although it has been known *for more than a century* [9, 10, 16] that such matrices of size $n\times n$ exist, so far there are no efficient numerical algorithms that can construct them. Even worse, it seems that the construction of such matrices might be an even harder problem than finding zeros of polynomials p and q. There exist simple and fast constructions [32, 40] that build matrices of size $\mathcal{O}(n^2)$ that satisfy (1.2), where the resulting two-parameter eigenvalue problem (1.3) is singular; we will discuss more details in section 4. Recent results [32] show that it is possible to solve singular two-parameter eigenvalue problems numerically for small to medium-sized matrices. However, the $\mathcal{O}(n^2)$ size of the matrices pushes the complexity of the algorithm to the enormous $\mathcal{O}(n^{12})$ and it is reported in [31] that this approach to compute zeros is competitive only for polynomials of degree $n<5$.

The construction of [32] yields matrices that are of asymptotic order $\frac{1}{2}n^2$, while those of [40] are of asymptotic order $\frac{1}{4}n^2$. In this paper we give two new representations. The first one uses the tree structure of monomials in x and y. The resulting matrices are smaller than those of [40], with the same asymptotic order $\frac{1}{4}n^2$. This representation can be used for bivariate polynomials as well as for polynomial two-parameter eigenvalue problems [33], that is, for polynomials with matrix coefficients. The second representation is even more condensed, with asymptotic order $\frac{1}{6}n^2$, and can be applied to scalar bivariate polynomials. Although the size of the matrices asymptotically still grows quadratically with n, the smaller size renders this approach attractive for polynomials of degree $n\lesssim 10$, or for larger n if the polynomials have only a few terms. This already is an interesting size for a practical use and might trigger additional interest in such methods that could culminate in even more efficient representations. Moreover, as we will see, for modest n, the order of the matrices is only roughly $2n$. Furthermore, for polynomials of degree 3, we present a new construction of matrices of order (exactly) 3.

There are other ways to study a system of polynomials as an eigenvalue problem (see, e.g., [11, 44]), but they involve more symbolic computation. In [28], an algorithm is proposed that only requires to solve linear systems and check rank conditions, which are similar tools that we use in the staircase method [32] to solve the obtained singular two-parameter eigenvalue problem. Of course, there are many numerical methods that can be applied to systems of bivariate polynomials. Two main approaches are the homotopy continuation and the resultant method; see, e.g., [12, 23, 41, 45, 49] and the references therein. There are also many methods which aim to compute only real solutions of a system of two real bivariate polynomials; see, e.g., [34, 43]. We compare our method with several existing approaches, including Mathematica's `NSolve` [51] and PHCpack [49], in section 7 and show that our approach is very competitive for polynomials up to degree $\lesssim 10$.

We mention that another advantage of writing the system of bivariate polynomials as a two-parameter eigenvalue problem is that then we can apply iterative subspace numerical methods such as the Jacobi–Davidson method and compute just a small part of zeros close to a given target (x_0, y_0) [19]; we will not pursue this approach in this paper.

The rest of this paper is organized as follows. In section 2 we give some applications where bivariate polynomial systems have to be solved. Section 3 introduces determinantal representations, and section 4 focuses on two-parameter eigenvalue problems. In section 5 we give a determinantal representation that is based on the "tree" of monomials, involves no computation, and is suitable for both scalar and matrix polynomials. The matrices of the resulting representation are asymptotically of order $\frac{1}{4}n^2$. In section 6 we give a representation with smaller matrices, of asymptotic order $\frac{1}{6}n^2$, that involves just a trivial amount of numerical computation (such as computing roots of low-degree univariate polynomials) and can be computed very efficiently. This representation may be used for scalar polynomials. We end with some numerical experiments in section 7 and conclusions in section 8.

2. Motivation.

Polynomial systems of form (1.1) arise in numerous applications and fields, such as signal processing [5, 8, 14, 47] and robotics [53]. In computer aided design, one may be interested in the intersections of algebraic curves, such as ellipses [2, 26, 29]. In two-dimensional subspace minimization [7], such as polynomial tensor optimization, one is interested in two-dimensional searches $\min_{\alpha,\beta} F(x + \alpha d_1 + \beta d_2)$, where $F : \mathbb{R}^n \to \mathbb{R}$, x is the current point, and d_1 and d_2 are search directions; see [42, 43] and the references therein.

In delay differential equations, determining critical delays in the case of so-called commensurate delays may lead to a problem of type (1.1) [22]. The simplest example is of the form $x'(t) = a\,x(t) + b\,x(t-\tau) + c\,x(t-2\tau)$, where $\tau > 0$ is the delay; asked are values of τ that result in periodic solutions. This yields p and q of degrees 2 and 3, respectively. Taking more delay terms with delays that are multiples of τ gives polynomials of higher degree.

In systems and control the first-order conditions of the L_2-approximation problem of minimizing $\|h-\widetilde{h}\|^2 = \int_0^\infty |h(t)-\widetilde{h}(t)|^2\,dt$, for a given impulse response h of degree n, and degree$(\widetilde{h}) = \widetilde{n} \le n$, lead to a system of type (1.1) [13].

When considering quadratic eigenvalue problems in numerical linear algebra, it is of interest to determine $\operatorname{argmin}_{\theta\in\mathbb{C}} \|(\theta^2 A + \theta B + C)u\|$ as an approximate eigenvalue for a given approximate eigenvector u, which gives a system of degree 3 in the real and imaginary part of θ [20, section 2.3]. Generalizations to polynomial eigenvalue problems give rise to polynomials p and q of higher degree.

Also, there has been some recent interest in this problem in the context of the chebfun2 project [34, 46]. In chebfun2, nonlinear real bivariate functions are approximated by bivariate polynomials, so solving (1.1) is relevant for finding zeros of systems of real nonlinear bivariate functions and for finding local extrema of such functions.

3. Determinantal representations.

In this section we introduce determinantal representations and present some existing constructions. The difference between what should theoretically be possible and what can be done in practice is huge. The algorithms we propose reduce the difference only by a small (but still significant) factor; there seems to be plenty of room for future improvements.

We say that a bivariate polynomial $p(x,y)$ has degree n if all its monomials $p_{ij}x^iy^j$ have total degree less than or equal to n, i.e., $i+j \le n$, and if at least one of the monomials has total degree equal to n. We say that the square $m\times m$ matrices A, B, and C form a *determinantal representation* of the polynomial p if $\det(A+xB+yC) = p(x,y)$. As our motivation is to use eigenvalue methods to solve polynomial systems, we will, instead of determinantal representation, often use the term *linearization* since a determinantal representation transforms an eigenvalue problem that involves polynomials of degree n into a linear eigenvalue problem (1.3). A definition of linearization that extends that for the univariate case (see, e.g., [27]) is the following.

DEFINITION 3.1. *A linear bivariate pencil $A+xB+yC$ of size $m\times m$ is a linearization of the polynomial $p(x,y)$ if there exist two polynomial matrices $L(x,y)$ and $Q(x,y)$ such that $\det(L(x,y)) \equiv \det(Q(x,y)) \equiv 1$ and*

$$L(x,y)\,(A+xB+yC)\,Q(x,y) = \begin{bmatrix} p(x,y) & 0 \\ 0 & I_{m-1}\end{bmatrix}.$$

We are interested not only in linearizations of scalar polynomials but also in linearizations of matrix bivariate polynomials of the form (cf. (1.1))

$$P(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{n-j} x^i\, y^j\, P_{ij}, \tag{3.1}$$

where the P_{ij} are $k\times k$ matrices. In line with the above, a linear pencil $A+xB+yC$ of matrices of size $m\times m$ presents a linearization (determinantal representation) of the matrix polynomial $P(x,y)$ if there exist two polynomial matrices $L(x,y)$ and $Q(x,y)$ such that $\det(L(x,y)) \equiv \det(Q(x,y)) \equiv 1$ and

$$L(x,y)\,(A+xB+yC)\,Q(x,y) = \begin{bmatrix} P(x,y) & 0 \\ 0 & I_{m-k}\end{bmatrix}.$$

In this case $\det(A+xB+yC) = \det(P(x,y))$. Each linearization of a matrix polynomial gives a linearization for a scalar polynomial, as we can think of scalars as of 1×1 matrices; the opposite is not true in general.

Dixon [10] showed that for every scalar bivariate polynomial $p(x,y)$ of degree n there exists a determinantal representation with symmetric matrices of size $n\times n$. Dickson [9] later showed that this result cannot be extended to general polynomials in more than two variables, except for three variables and polynomials of degree two and three, and four variables and polynomials of degree two. Although they both give constructive proofs, there does not seem to exist an efficient numerical algorithm to construct the determinantal representation with matrices of size $n \times n$ for a given bivariate polynomial of degree n.

In recent years, the research in determinantal representations has been growing, as determinantal representations for a particular subset of polynomials, *real zero polynomials*, are related to linear matrix inequality (LMI) constraints used in semidefinite programming (SDP). For an overview see, e.g., [36, 50]; here we give just the essentials for bivariate polynomials that are related to our problem.

We say that a real polynomial $p(x, y)$ satisfies the *real zero condition* with respect to $(x_0, y_0) \in \mathbb{R}^2$ if for all $(x, y) \in \mathbb{R}^2$ the univariate polynomial $p_{(x,y)}(t) = p(x_0 + tx, y_0 + ty)$ has only real zeros. A two-dimensional *LMI set* is defined as

$$\left\{(x, y) \in \mathbb{R}^2 : A + xB + yC \succeq 0\right\},$$

where A, B, and C are symmetric matrices of size $m \times m$ and $\succeq 0$ stands for positive semidefinite. In SDP we are interested in convex sets $\mathcal{S} \subset \mathbb{R}^2$ that admit an LMI representation, i.e., $\mathcal{S}$ is an LMI set for certain matrices A, B, and C. Such sets are called *spectrahedra* and Helton and Vinnikov [16] showed that such $\mathcal{S}$ must be an algebraic interior, whose minimal defining polynomial p satisfies the real zero condition with respect to any point in the interior of $\mathcal{S}$. Their results state that if a polynomial $p(x, y)$ of degree n satisfies real zero condition with respect to (x_0, y_0), then there exist symmetric matrices A, B, and C of size $n \times n$ such that $\det(A + xB + yC) = p(x, y)$ and $A + x_0B + y_0C \succeq 0$. Matrices A, B, and C thus form a particular determinantal representation for p.

The problem of constructing an LMI representation with symmetric or Hermitian matrices A, B, and C for a given spectrahedron $\mathcal{S}$ raised much more interest than the related problem of generating a determinantal representation for a generic bivariate polynomial. There exist procedures, which rely heavily on slow symbolic computation or other expensive steps, that return an LMI representation with Hermitian matrices for a given spectrahedron, but they are not efficient enough. For instance, a method from [38], based on the proof from [10], does return $n \times n$ matrices for a polynomial of degree n, but the reported times (10 seconds for a polynomial of degree 10) show that it is much too slow for our purpose. As a first step of the above method is to find zeros of a system of bivariate polynomials of degree n and $n - 1$, this clearly cannot be efficient enough for our needs. In addition, we are interested in determinantal representations for polynomials that do not necessary satisfy the real zero condition.

In SDP and LMI the matrices have to be symmetric or Hermitian, which is not required in our case. We need a simple and fast numerical construction of matrices that satisfy (1.2) and are as small as possible—ideally their size should increase linearly and not quadratically with n.

Regarding available determinantal representations for generic bivariate polynomials, we first have the linearization by Khazanov with matrices of size $n^2 \times n^2$ [25]. In [33, Appendix], a smaller linearization for bivariate matrix polynomials is given with block matrices of order $\frac{1}{2}n(n + 1)$. The linearization uses all monomials of degree up to $n - 1$ and contains a direct expression for the matrices A, B, and C such that $\det(A + xB + yC) = p(x, y)$. Similar to [25], it can be applied to matrix polynomials. We give an example for a general matrix polynomial of degree 3, from which it is possible to deduce the construction for a generic degree. This linearization will be superseded in section 5 by a more economical one.

Example 3.2 (see [33, Appendix]). We take a matrix bivariate polynomial of degree 3,

$$P(x,y) = P_{00}+xP_{10}+yP_{01}+x^2P_{20}+xyP_{11}+y^2P_{02}+x^3P_{30}+x^2yP_{21}+xy^2P_{12}+y^3P_{03}.$$

If u is a nonzero vector, then $P(x,y)u=0$ if and only if $(A+xB+yC)\underline{u}=0$, where

$$A+xB+yC = \begin{bmatrix} P_{00} & P_{10} & P_{01} & P_{20}+xP_{30} & P_{11}+xP_{21} & P_{02}+xP_{12}+yP_{03} \\ -xI_k & I_k & 0 & 0 & 0 & 0 \\ -yI_k & 0 & I_k & 0 & 0 & 0 \\ 0 & -xI_k & 0 & I_k & 0 & 0 \\ 0 & 0 & -xI_k & 0 & I_k & 0 \\ 0 & 0 & -yI_k & 0 & 0 & I_k \end{bmatrix} \tag{3.2}$$

and

$$\underline{u} = u \otimes [1 \quad x \quad y \quad x^2 \quad xy \quad y^2]^T.$$

We have $\det(A+xB+yC) = \det(P(x,y))$ and $A+xB+yC$ is a linearization of $P(x,y)$.

We remark that Quarez [40] also gives explicit expressions for determinantal representations. He is interested in symmetric representations and is able to construct, for a bivariate polynomial of degree n such that $p(0,0) \neq 0$, a linearization with symmetric matrices of size $N \times N$, where

$$N = 2\binom{\lfloor n/2 \rfloor + 2}{2} \approx \frac{n^2}{4}. \tag{3.3}$$

This has asymptotically the same order as the first linearization that we will give in section 5; the second linearization in section 6 has a smaller order. We also remark that in the phase, when we are solving a two-parameter eigenvalue problem to compute the zeros of a system of two bivariate polynomials, we cannot exploit the fact that the matrices are symmetric, so this is not important for our application.

There are some other available tools, for instance, it is possible to construct a determinantal representation using the package NCAlgebra for noncommutative algebra [15, 35] that runs in Mathematica [51], but this does not give satisfactory results for our application as the matrices that we can construct have smaller size.

4. Two-parameter eigenvalue problems.

In this section we briefly present the two-parameter eigenvalue problem and some available numerical methods. A motivation for the search for small determinantal representations is that if we transform a system of bivariate polynomials into an eigenvalue problem, then we can apply existing numerical methods for such problems.

A two-parameter eigenvalue problem has the form (1.3), where A_i, B_i, and C_i are given $n_i \times n_i$ complex matrices. We are looking for $x, y \in \mathbb{C}$ and nonzero vectors $u_i \in \mathbb{C}^{n_i}$, $i=1,2$, such that (1.3) is satisfied. In such case we say that a pair (x,y) is an eigenvalue and the tensor product $u_1 \otimes u_2$ is the corresponding eigenvector. If we introduce the so-called operator determinants, the matrices

$$\begin{aligned} \Delta_0 &= B_1 \otimes C_2 - C_1 \otimes B_2, \\ \Delta_1 &= C_1 \otimes A_2 - A_1 \otimes C_2, \\ \Delta_2 &= A_1 \otimes B_2 - B_1 \otimes A_2, \end{aligned} \tag{4.1}$$

then the problem (1.3) is related to a coupled pair of generalized eigenvalue problems

$$\begin{aligned} \Delta_1 w &= x\ \Delta_0 w, \\ \Delta_2 w &= y\ \Delta_0 w \end{aligned} \tag{4.2}$$

for a decomposable tensor $w = u_1 \otimes u_2$. If Δ_0 is nonsingular, then Atkinson [1] showed that the solutions of (1.3) and (4.2) agree and the matrices $\Delta_0^{-1}\Delta_1$ and $\Delta_0^{-1}\Delta_2$ commute. In the nonsingular case the two-parameter problem (1.3) has $n_1 n_2$ eigenvalues and we can numerically solve it with a variant of the QZ algorithm on (4.2) from [17]. Ideally, if we could construct a determinantal representation with matrices $n \times n$ for a bivariate polynomial of degree n, this would be the method that we would apply on the "companion" two-parameter eigenvalue problem to get the zeros of the polynomial system. As Δ_0, Δ_1, and Δ_2 have size $n_1 n_2 \times n_1 n_2$, the computation of all eigenvalues of a nonsingular two-parameter eigenvalue problem has time complexity $\mathcal{O}(n_1^3\, n_2^3)$, which would lead to $\mathcal{O}(n^6)$ algorithm for a system of bivariate polynomials. Of course, for this approach we need a construction of a determinantal representation with matrices $n \times n$ that should not be more computationally expensive than the effort to solve a two-parameter eigenvalue problem.

Unfortunately, all practical constructions for determinantal representations (including the two presented in this paper) return matrices that are much larger than $n \times n$. If we have a determinantal representation with matrices larger than the degree of the polynomial, then the corresponding two-parameter eigenvalue problem is singular, which means that both matrix pencils (4.2) are singular, and we are dealing with a more difficult problem. There exists a numerical method from [33] that computes the regular eigenvalues of (1.3) from the common regular part of (4.2). For the generic singular case it is shown in [32] that the regular eigenvalues of (1.3) and (4.2) agree. For other types of singular two-parameter eigenvalue problems the relation between the regular eigenvalues of (1.3) and (4.2) is not completely known, but the numerical examples indicate that the method from [33] can be successfully applied to such problems as well. However, the numerical method, which is a variant of a staircase algorithm [48], has to make a lot of decisions on the numerical rank and a single inaccurate decision can cause the method to fail. As the size of the matrices increases, the gaps between singular values may numerically disappear and it may be difficult to solve the problem.

This is not the only issue that prevents the use of determinantal representations to solve a bivariate system. The algorithm for the singular two-parameter eigenvalue problems still has complexity $\mathcal{O}(n_1^3\, n_2^3)$, but the fast determinantal representations that we are aware of return matrices of size $\mathcal{O}(n^2)$ instead of $\mathcal{O}(n)$. This is what pushes the overall complexity to $\mathcal{O}(n^{12})$ and makes this approach efficient only for polynomials of small degree. Nonetheless, at complexity so high, each construction that gives a smaller determinantal representation can make a change. In view of this, we propose two new linearizations in the next two sections.

5. First linearization.

We are interested in linearizations of the matrix polynomial

$$P(x,y) = P_{00} + xP_{10} + yP_{01} + \cdots + x^n P_{n0} + x^{n-1} y P_{n-1,1} + \cdots + y^n P_{0n}$$

of degree n, where P_{ij} are square matrices. Our goal is to find square matrices A, B, and C as small as possible such that $\det(A + xB + yC) = \det(P(x,y))$. Also, we need a relation that $P(x,y)u = 0$ if and only if $(A + xB + yC)\underline{u} = 0$, where $\underline{u}$ is a tensor product of u and a polynomial of x and y. The linearization in this section also applies to scalar bivariate polynomials, where all matrices are 1×1 and $u = 1$.

In section 3 we have given a linearization with block matrices of order $\frac{1}{2}n(n+1)$. We can view this linearization in the following way. If $P(x,y)u = 0$ for $u \neq 0$, then $\det(A + xB + yC)\underline{u} = 0$, where the vector $\underline{u}$ has the form

$$\underline{u} = u \otimes [1 \quad x \quad y \quad x^2 \quad xy \quad y^2 \quad \cdots \quad x^{n-1} \quad x^{n-2}y \quad \cdots \quad y^{n-1}]^T. \tag{5.1}$$

This means that $\underline{u}$ always begins with the initial block u and then contains all blocks of the form $x^j y^k u$, where $j + k \leq n - 1$. To simplify the presentation we will usually omit u when referring to the blocks of the vector (5.1). The blocks are ordered in the degree negative lexicographic ordering, i.e., $x^a y^b \prec x^c y^d$ if $a+b < c+d$, or $a+b = c+d$ and $a > c$.

The above block structure of vector $\underline{u}$ is defined in the rows of the matrix from the second one to the last one (see Example 3.2). For each block $s = x^j y^k$ of (5.1) such that $j + k \geq 1$ there always exists a preceding block q of the grade $j + k - 1$ such that either $s = xq$ or $s = yq$ (when $j \geq 1$ and $k \geq 1$ both options are possible). Suppose that $s = xq$, $\mathrm{ind}(s) = i_s$, and $\mathrm{ind}(q) = i_q$, where function ind returns the index of a block. Then the matrix $A + xB + yC$ has block $-xI$ on position (i_s, i_q) and block I on position (i_s, i_s). These are the only nonzero blocks in the block row i_s. A similar construction with $-xI$ replaced by $-yI$ is used in the case $s = yq$.

The first block row of the matrix $A + xB + yC$ is used to represent the matrix polynomial $P(x,y)$. One can see that there exist linear pencils $A_{1i} + xB_{1i} + yC_{1i}$, $i = 1, \ldots, m$, such that

$$P(x,y)u = \begin{bmatrix} A_{11} + xB_{11} + yC_{11} & \cdots & A_{1m} + xB_{1m} + yC_{1m} \end{bmatrix} \underline{u}, \tag{5.2}$$

where $m = \frac{1}{2}n(n+1)$ is the number of blocks in (5.1). The pencils in (5.2) are not unique. For instance, a term $x^j y^k P_{jk}$ of $P(x,y)$ can be represented in one of up to three possible ways:

(a) if $j + k < n$, we can set $A_{1p} = P_{jk}$, where $p = \mathrm{ind}(x^j y^k)$,
(b) if $j > 0$, we can set $B_{1p} = P_{jk}$, where $p = \mathrm{ind}(x^{j-1} y^k)$,
(c) if $k > 0$, we can set $C_{1p} = P_{jk}$, where $p = \mathrm{ind}(x^j y^{k-1})$.

Based on the above discussion we see that not all the blocks in (5.1) are needed to represent a matrix polynomial $P(x,y)$. What we need is a minimal set of monomials $x^j y^k$, where $j + k < n$, that is sufficient for a matrix polynomial of degree n. We can formulate the problem of finding the smallest possible set for a given polynomial as a graph problem.

We can think about all possible terms $x^j y^k$, where $j + k < n$, as of nodes in a directed graph G with the root 1 and a directed edge from node s to node t if $t = xs$ or $t = ys$ (see Figure 1 for the case $n = 5$). Now, we are looking for the smallest connected subgraph G' with a root 1 that can represent a given polynomial. Equivalently, we are looking for a minimal directed rooted tree. Let us remember that for each term $x^j y^k P_{jk}$ of the polynomial $P(x,y)$ there are up to three possible nodes in the graph G that can be used to represent it. It is sufficient that one of these nodes is in a minimal tree G'. Furthermore, if $j + k > 1$, then we can assume that we always use a node of degree $j + k - 1$ to represent $x^j y^k P_{jk}$ and then there are only one or two options for a given term. All together, each nonzero term $x^j y^k P_{jk}$, where $j + k > 0$, in the polynomial P defines one of the following rules for the subgraph G':

(a) if $k = 0$, then $x^{j-1} y^0$ has to be in the subgraph G',
(b) if $j = 0$, then $x^0 y^{k-1}$ has to be in the subgraph G',

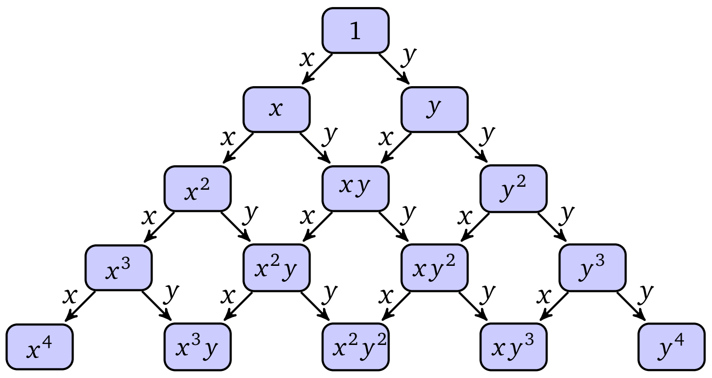

FIG. 1. *Graph G for the polynomial of degree* 5.

(c) if $j > 0$, and $k > 0$, then at least one of $x^{j-1}y^k$ or x^jy^{k-1} has to be in the subgraph G'.

The term P_{00} can be presented by the root 1, which is always present in the subgraph G'.

Finding a minimal tree for a given polynomial is not an easy problem: it can be formulated as an NP-hard directed Steiner tree problem (DST) (see, e.g., [24]), where one has a directed graph $G = (V, E)$ with nonnegative weights on edges and the goal is to find the minimum weight directed rooted tree that connects all terminals $X \subset V$ to a given root $r \in V$.

Suppose that we are looking for a minimal representation tree for a polynomial $P(x, y)$ of degree n. In the graph G, which contains all nodes x^jy^k for $j + k < n$ (see Figure 1 for the case $n = 5$), we put weight 1 on all directed x and y edges. Now we add a new vertex for each monomial x^jy^k that is present in $P(x, y)$ and connect it with zero weight edges from all possible nodes in G that could be used to represent the monomial in the linearization. We make a DST problem by taking node 1 as a root and all newly added vertices as terminals. From a solution of the DST problem the minimal representation tree can be recovered. Although this is an NP-hard problem, there exist some polynomial time approximation algorithms that give a solution close to the optimal one and could be used to construct a small determinantal representation for a given polynomial with small number of nonzero terms. For the latest available algorithms, see, e.g., [4] and the references therein.

Example 5.1. We are interested in a minimal tree for the matrix polynomial

$$P(x, y) = P_{00} + xP_{10} + yP_{01} + y^3P_{03} + x^2y^2P_{22} + x^4yP_{41} + xy^4P_{14} + x^6P_{60} + x^2y^4P_{24}. \tag{5.3}$$

Nonzero terms in (5.3) define the nodes that have to be present in the minimal subgraph. They either are strictly defined as are the nodes 1, y^2, and x^5 or come in pairs where at least one element of each pair has to be present in the subgraph. Such pairs are $(x^2y, xy^2), (x^4, x^3y)$, and (x^2y^3, xy^4). The situation is presented in Figure 2, where nodes and pairs, such that either the left or the right node has to be included, are shadowed in green. The nodes of the minimal connected subgraph that includes all required nodes are colored red.

In a DST formulation each green shadow presents a terminal linked by zero weight edges to one or two nodes that are included in the region. On all other edges we put weight 1 and then search for the minimum weight directed rooted tree that connects all terminals to the root 1.

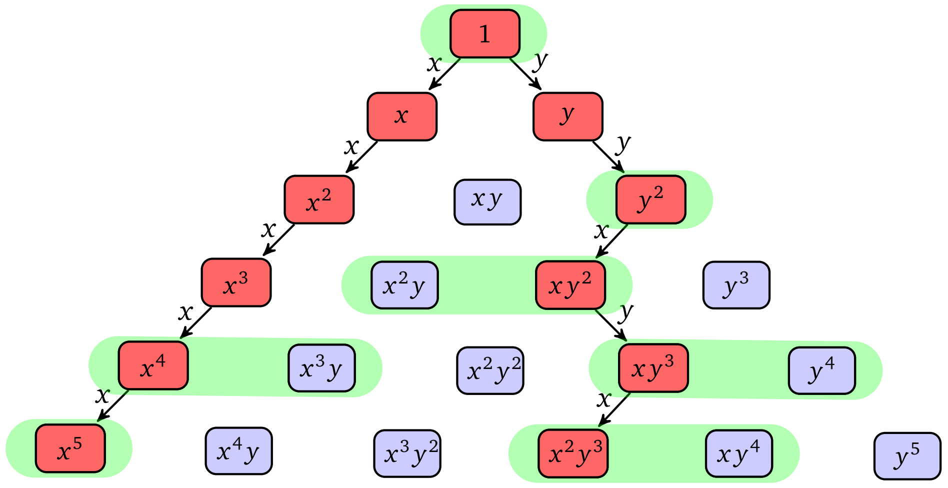

FIG. 2. *A minimal directed subtree G' for the matrix polynomial* (5.3) *of degree* 6.

Matrix polynomial (5.3) can thus be represented with matrices of block size 11×11. If we order the nodes of the subgraph in the degree negative lexicographic ordering, then $\underline{u}$ has the form

$$\underline{u} = u \otimes [1 \quad x \quad y \quad x^2 \quad y^2 \quad x^3 \quad xy^2 \quad x^4 \quad xy^3 \quad x^5 \quad x^2y^3]^T$$

and a possible first block row of $A + xB + yC$ has the form

$$\begin{bmatrix} P_{10} + xP_{10} + yP_{01} & 0 & 0 & 0 & yP_{03} & 0 & yP_{22} & yP_{41} & yP_{14} & xP_{60} & yP_{24} \end{bmatrix}.$$

In the subsequent block rows, the matrix $A+xB+yC$ has only 20 nonzero blocks, 10 of them identity blocks on the main diagonal. The remaining nonzero blocks are $-xI$ on block positions $(2,1)$, $(4,2)$, $(6,4)$, $(7,5)$, $(8,6)$, $(10,8)$, $(11,9)$ and blocks $-yI$ on positions $(3,1), (5,3), (9,7)$.

If we have a generic matrix polynomial $P(x,y)$, whose terms are all nonzero, then it is easy to see that the subgraph that contains all terms x^jy^k, where $j+k<n$ and either $j=0$ or k is even, is minimal. The detailed situation for the case $n=6$ is presented in Figure 3, and representation trees for polynomials of degree 1 to 8 are presented in Figure 4. Counting the number of nodes in the tree gives the following result:

$$\psi(n) := |G'| = \begin{cases} \frac{1}{4}n(n+1), & n \text{ even}, \\ \frac{1}{4}(n-1)(n+5)+1, & n \text{ odd}. \end{cases} \tag{5.4}$$

If we compare this with the linearization from Example 3.2 that has matrices of block size $\frac{1}{2}n(n+1)$, we see that the new linearization uses matrices of roughly half size. The size of the matrices is also smaller than (3.3) from [40], while having the same asymptotic order.

THEOREM 5.2. *We can linearize each matrix polynomial $P(x,y)$ of degree n with matrices of block size $\psi(n)$ from* (5.4) *using a minimal tree G' that contains the terms x^jy^k, where $j+k<n$ and either $j=0$ or k is even.*

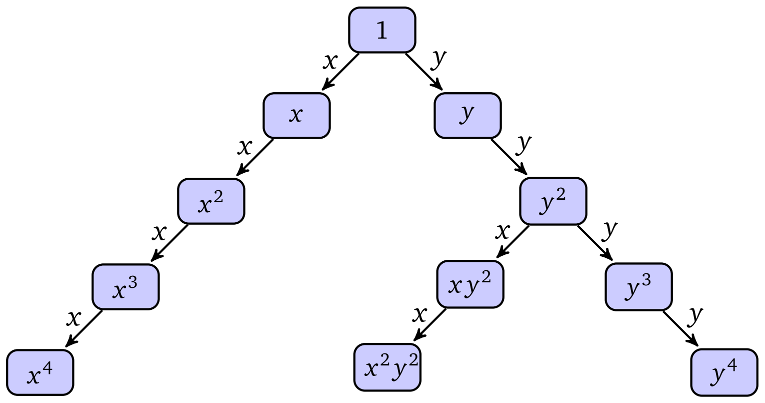

FIG. 3. *A minimal tree G' for a generic polynomial of degree* 5.

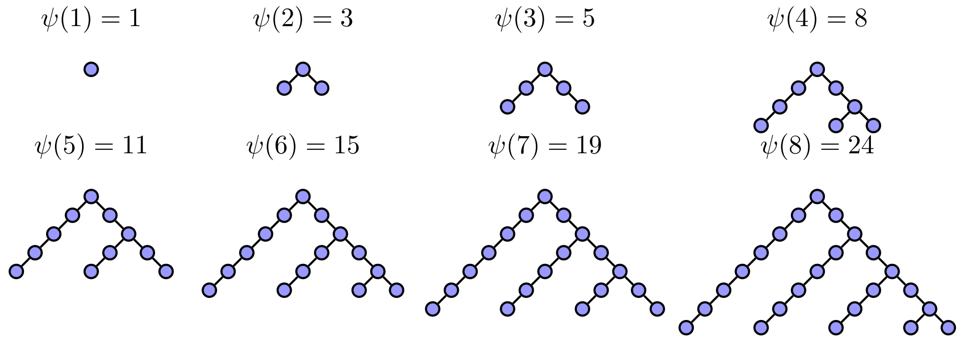

FIG. 4. *Minimal representation trees for polynomials of degrees* 1 *to* 8.

Proof. We order all nodes of a minimal tree G' in the degree negative lexicographic ordering and form the block matrix $L(x, y)$ in the following way. All diagonal blocks of $L(x, y)$ are I. If a node with index p is connected to a node with index q with an x or y edge, then we put $-xI$ or $-yI$ in the block position (q, p), respectively. Because of the ordering, the matrix L is block lower triangular and nonsingular. Its inverse $L(x, y)^{-1}$ is therefore also a lower triangular matrix with diagonal identity blocks.

Let $m = \psi(n)$ be the number of nodes in G'. If follows from $L(x, y)L(x, y)^{-1} = I$ that the first block column of $L(x, y)^{-1}$ has the form

$$I \otimes \begin{bmatrix} 1 & s_2 & s_3 & \cdots & s_m \end{bmatrix}^T, \tag{5.5}$$

where s_j is the monomial in the jth node of G' for $j = 1, \ldots, m$ $(s_1 = 1)$.

Now we will construct the linearization of the matrix polynomial $P(x, y)$. We need a block matrix $M(x, y) = A + xB + yC$, whose elements are linear pencils in x and y. We take $M(x, y) = L(x, y)$ and adjust the first block row $M_1(x, y)$, where we put linear pencils such that

$$M_1(x, y)(I \otimes \begin{bmatrix} 1 & s_2 & s_3 & \cdots & s_m \end{bmatrix}^T) = P(x, y).$$

This is always possible as for each term $x^j y^k P_{jk}$ in the polynomial $P(x, y)$ there exists a term $x^r y^q$ in G' such that $(j, k) - (r, q)$ is one of the following three options: $(0, 0)$, $(1, 0)$, or $(0, 1)$. The product $M(x, y)L(x, y)^{-1}$ is an upper block triangular matrix of the form

$$M(x,y)L(x,y)^{-1} = \begin{bmatrix} P(x,y) & H_2(x,y) & \cdots & H_m(x,y) \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix},$$

where $H_2(x,y),\dots,H_m(x,y)$ are matrix polynomials. If we introduce the matrix polynomial

$$U(x,y) = \begin{bmatrix} I & -H_2(x,y) & \cdots & -H_m(x,y) \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix},$$

then it follows that

$$U(x,y)\,M(x,y)\,L(x,y)^{-1} = \begin{bmatrix} P(x,y) & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix},$$

and since $\det(L(x,y)) \equiv \det(U(x,y)) \equiv 1$, this proves that $M(x,y) = A + xB + yC$ is indeed a linearization of the matrix polynomial $P(x,y)$. ☐

Example 5.3. As an example we consider the scalar bivariate polynomial

$$p(x,y) = 1 + 2x + 3y + 4x^2 + 5xy + 6y^2 + 7x^3 + 8x^2y + 9xy^2 + 10y^3,$$

which was already linearized in [32] with matrices of size 6×6 (we can also get a 6×6 linearization if we insert the coefficients in matrix (3.2) of Example 3.2). Now we can linearize it with matrices of size 5×5 as $p(x,y) = \det(A + xB + yC)$, where

$$A + xB + yC = \begin{bmatrix} 1+2x+3y & 4x+5y & 6y & 7x+8y & 9x+10y \\ -x & 1 & 0 & 0 & 0 \\ -y & 0 & 1 & 0 & 0 \\ 0 & -x & 0 & 1 & 0 \\ 0 & 0 & -y & 0 & 1 \end{bmatrix}.$$

In the next section we will further reduce the size of the matrices to 4×4 and even 3×3.

6. Second linearization.

We will upgrade the approach from the previous section and produce even smaller representations for scalar polynomials. As before, representations have a form of the directed tree, but instead of using only x and y, an edge can now be any linear polynomial $\alpha x + \beta y$ such that $(\alpha,\beta) \neq (0,0)$. These additional parameters give us enough freedom to produce smaller representations. The root is still 1, while the other nodes are polynomials in x and y that are products of all edges on the path from the root to the node. In each node all monomials have the same degree, which is equal to the graph distance to the root. Before we continue with the construction, we give a small example to clarify the idea.

Example 6.1. A linearization of a polynomial of degree 3 with matrices of size 4×4 is presented in Figure 5. We now explain the figure and show how to produce

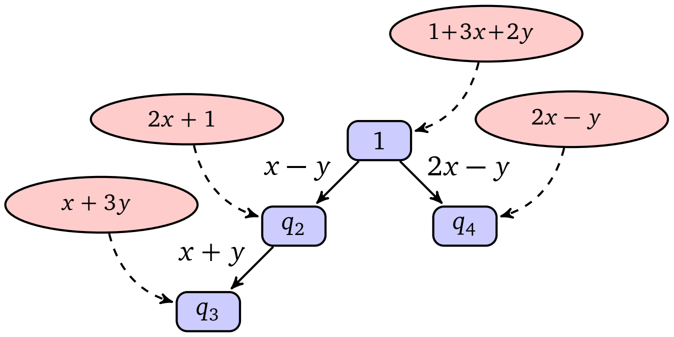

FIG. 5. *A representation tree and a linearization for a polynomial of degree* 3.

the matrices from the representation tree. The nodes in the representation tree are the following polynomials:

$$q_1(x,y) = 1, \qquad q_2(x,y) = (x-y)\,q_1(x,y) = x-y,$$
$$q_3(x,y) = (x+y)\,q_2(x,y) = x^2-y^2, \qquad q_4(x,y) = (2x-y)\,q_1(x,y) = 2x-y.$$

The polynomial of degree 3 is then a linear combination of nodes in the representation tree and coefficients $f_1,\dots,f_4$ which are polynomials of degree 1 contained in the ellipses. This gives

$$\begin{aligned} p(x,y) &= (1+3x+2y)\,q_1(x,y) + (2x+1)\,q_2(x,y) + (x+3y)\,q_3(x,y) \\ &\quad + (2x-y)\,q_4(x,y) \\ &= 1+4x+y+6x^2-6xy+y^2+x^3+3x^2y-xy^2-3y^3. \end{aligned}$$

Similarly as in section 5, we can write the matrices by putting the linear coefficients in the first row and relations between the polynomials $q_1(x,y)$ to $q_4(x,y)$ in the subsequent rows. For each edge of the form $q_k = (\alpha x+\beta y)\,q_j$ we put $-(\alpha x+\beta y)$ in the position (k,j) in the matrix $M(x,y) = A + xB + yC$ and 1 in the position (k,k). In the first row we put $a_k+b_kx+c_ky$ in the position $(1,k)$ if $f_k(x,y) = a_k+b_kx+c_ky$ is the linear factor that multiplies the polynomial $q_k(x,y)$ in the linearization. The matrix $M(x,y)$ that corresponds to Figure 5, such that $\det(M(x,y)) = p(x,y)$, is

$$M(x,y) = \begin{bmatrix} 1+3x+2y & 2x+1 & x+3y & 2x-y \\ -x+y & 1 & 0 & 0 \\ 0 & -x-y & 1 & 0 \\ -2x+y & 0 & 0 & 1 \end{bmatrix}.$$

In Example 6.1 we showed how to construct the bivariate pencil $M(x,y) = A + xB + yC$ from a representation tree and the corresponding linear coefficients. The outline of an algorithm that constructs a representation tree and the corresponding linear coefficients for a given polynomial $p(x,y)$ is presented in Algorithm 1. In the following discussion we give some missing details and show that the algorithm indeed gives a linearization.

- The nodes $q_2,\dots,q_n$ that we construct in step 2 are polynomials of the form $q_k(x,y) = (x-\zeta_1 y)\cdots(x-\zeta_{k-1}y)$ for $k = 2,\dots,n$. All monomials in q_k have degree $k-1$ and the leading term is x^{k-1}.

Algorithm 1. Given a bivariate polynomial $p(x,y) = \alpha_{00}+\alpha_{10}x+\alpha_{01}y+\cdots+\alpha_{n0}x^n+\alpha_{n-1,1}x^{n-1}y+\cdots+\alpha_{0n}y^n$ such that $\alpha_{n0} \neq 0$, the algorithm returns a representation tree with a determinantal representation of the polynomial.

```
1. Compute $n$ zeros $\zeta_1,\dots,\zeta_n$ of the polynomial $h(t) = \alpha_{n0}t^n + \alpha_{n-1,1}t^{n-1} + \cdots + \alpha_{0n}$.
2. Form a branch of the tree with the root $q_1(x,y) \equiv 1$ and nodes $q_2,\dots,q_n$,
   where $q_{k+1}$ is a successor of $q_k$ and the edge from $q_k$ to $q_{k+1}$ contains the
   factor $x - \zeta_k y$ for $k = 1,\dots,n-1$.
3. Compute linear coefficients $f_1,\dots,f_n$ for nodes $q_1,\dots,q_n$ in the following way:
   (a) take $f_1(x,y) = \alpha_{00} + \alpha_{10}x + \alpha_{01}y$,
   (b) take $f_k(x,y) = \alpha_{k0}x + (\alpha_{k-1,1} - \alpha_{k0}\beta_k)y$, where $\beta_k$ is a coefficient of
       $q_k(x,y)$ at $x^{k-1}y$, for $k = 2,\dots,n-1$,
   (c) take $f_n(x,y) = \alpha_{n0}(x - \zeta_n y)$.
4. Compute the remainder $r(x,y) = p(x,y) - \sum_{i=1}^n f_k(x,y)\,q_k(x,y)$, which has
   the form $r(x,y) = y^2 s(x,y)$, where $s(x,y)$ is a polynomial of degree $n-3$.
5. If $s(x,y) \equiv 0$, then stop and return the tree.
6. Add node $q_{n+1}$ and connect it to the root by an edge having the factor $y$.
7. If $s(x,y)$ is a nonzero constant $\beta_{00}$, then use $f_{n+1} = \beta_{00}y$ as a coefficient for
   the node $q_{n+1}$, stop, and return the tree.
8. Recursively call the same algorithm to obtain a representation tree with the
   root $q_1'$ for the polynomial $s(x,y)$.
9. Connect $q_{n+1}$ to $q_1'$ by an edge with a factor $y$ and return the tree with the
   root $q_1$.
```

- Each product $q_k(x,y)f_k(x,y)$ for $k = 2,\dots,n$ is a polynomial with monomials of exact degree k, while $q_1(x,y)f_1(x,y)$ is a polynomial of degree 1. The linear factors $f_k(x,y)$ in step 3 are constructed so that
  - The leading two monomials (x^k and $x^{k-1}y$) of $f_k(x,y)\,q_k(x,y)$ agree with the part $\alpha_{k0}x^k + \alpha_{k-1,1}x^{k-1}y$ of the polynomial $p(x,y)$ for $k = 2,\dots,n-1$,
  - the product $f_n(x,y)\,q_n(x,y) = a_{n0}(x-\zeta_1 y)\cdots(x-\zeta_n y)$ agrees with the part of $p(x,y)$ composed of all monomials of degree exactly n,
  - the product $q_1(x,y)\,f_1(x,y) = a_{00} + a_{10}x + a_{01}y$ agrees with the part of $p(x,y)$ composed of all monomials of degree up to 1.

  As a result, the remainder in step 4 has the form $y^2 s(x,y)$, where $s(x,y)$ is a polynomial of degree $n-3$. The situation at the end of step 4 is presented in Figure 6.
- If the coefficient α_{n0} is zero, then we can apply a linear substitution of x and y of the form $x = \widetilde{x}$ and $y = \widetilde{y} + \gamma\widetilde{x}$, where we pick γ such that

$$\alpha_{n-1,1}\,\gamma + \alpha_{n-2,2}\,\gamma^2 + \cdots + \alpha_{0n}\,\gamma^n \neq 0.$$

  This ensures that the substituted polynomial in $\widetilde{x}$ and $\widetilde{y}$ will have a nonzero coefficient at $\widetilde{x}^n$. After we complete the representation tree for the substituted polynomial in $\widetilde{x}$ and $\widetilde{y}$, we perform the substitution back to x and y.
- If the polynomial $s(x,y)$ in step 4 is not a constant, then we obtain a representation subtree for $s(x,y)$ by calling recursively the same algorithm. In order to obtain the final representation tree, we then join the existing branch

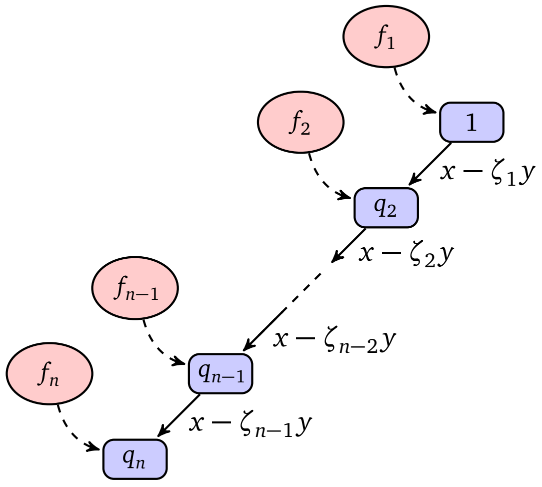

Fig. 6. *The representation tree after step 4 of Algorithm 1. The remainder $p(x,y) - \sum_{j=1}^{n} f_j(x,y)\, q_j(x,y)$ is a polynomial of the form $y^2 s(x,y)$, where $s(x,y)$ is a polynomial of degree $n-3$.*

to the representation subtree for the polynomial $s(x,y)$. We do this by introducing a new node q_{n+1} in step 6 that is linked to the root by the edge with the factor y. To this new node we link the root q_1' of the subtree for the polynomial $s(x,y)$ in step 9, again using the edge with the factor y. As q_1' is linked to the root by two edges y, this multiplies all nodes in the subtree by y^2 and, since the subtree is a representation for $s(x,y)$, this gives a representation for the remainder $r(x,y)$ from step 4. The situation after step 9 with the final representation tree for the polynomial $p(x,y)$ is presented in Figure 7.

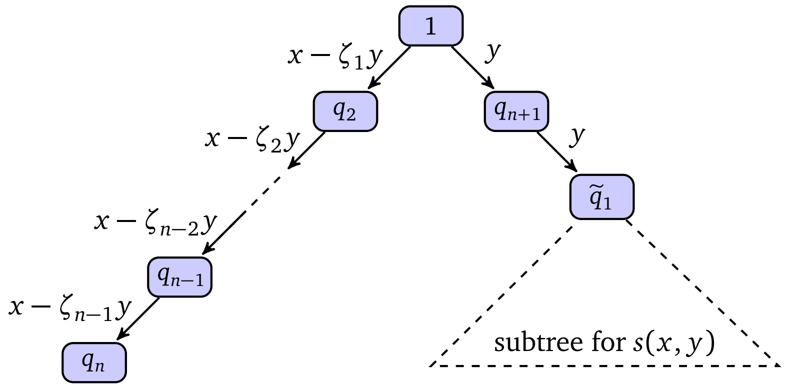

Fig. 7. *The final representation tree.*

From the output of Algorithm 1, matrices A, B, C such that $\det(A+xB+yC) = p(x,y)$ can be obtained in the same way as in Example 6.1. We remark that the zeros $\zeta_1, \ldots, \zeta_n$ in step 1 can be complex, even if the polynomial p has real coefficients. Thus, in a general case a linearization produced by Algorithm 1 has complex matrices A, B, and C.

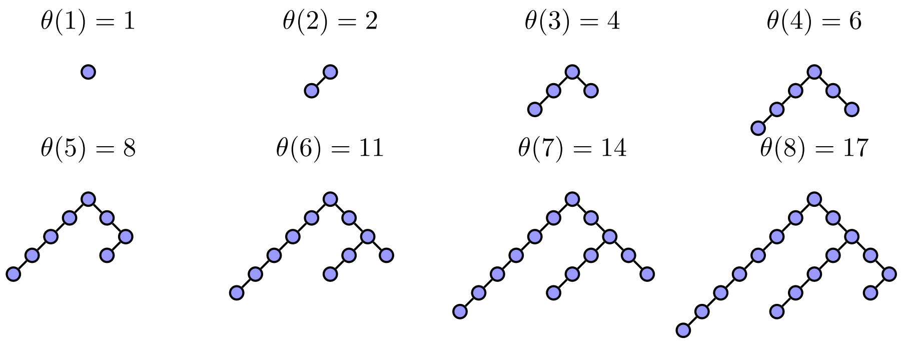

FIG. 8. *Representation trees for polynomials of degrees* 1 *to* 8.

Example 6.2. We apply Algorithm 1 on $p(x,y) = 1+2x+3y+4x^2+5xy+6y^2+7x^3+8x^2y+9xy^2+10y^3$ from Example 5.3. First, we compute the roots

$$\zeta_1 = -0.0079857 - 1.1259i, \quad \zeta_2 = -0.0079857 + 1.1259i, \quad \zeta_3 = -1.1269 \tag{6.1}$$

of the polynomial $h(t) = 7t^3 + 8t^2 + 9t + 10$. The zeros are ordered so that $|\zeta_1| \le |\zeta_2| \le \cdots \le |\zeta_n|$. In exact computation the order is not important, but in numerical tests we experience better results with this order. This gives the polynomials in the first branch of the representation tree:

$$\begin{aligned} q_1(x,y) = 1, \quad q_2(x,y) &= x + (0.0079857 + 1.1259i)y, \\ q_3(x,y) &= x^2 + 0.015971xy + 1.2677y^2, \end{aligned}$$

and we can compute the corresponding coefficients

$$f_1(x,y) = 1+2x+3y, \quad f_2(x,y) = 4x+(4.9681+4.5036i)y, \quad f_3(x,y) = 7x+7.8882y.$$

For the remainder $r(x,y) = p(x,y) - \sum_{j=1}^{3} f_j(x,y)\, q_j(x,y) = (0.88972+5.5576i)y^2$ we need just one additional node $q_4(x,y) = y$ with the coefficient $f_4(x,y) = (0.88972 + 5.5576i)y$. The determinantal representation with 4×4 matrices is $p(x,y) = \det(A + xB + yC)$, where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 4 & 7 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \text{and}$$

$$C = \begin{bmatrix} 3 & 4.9681 + 4.5036i & 7.8882 & 0.88972 + 5.5576i \\ -0.0079857 + 1.1259i & 0 & 0 & 0 \\ 0 & -0.0079857 - 1.1259i & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}.$$

Representation trees for polynomials of degree 1 to 8 are presented in Figure 8. If we compare them to the determinantal representations from section 5 in Figure 4, then we see that representations obtained by Algorithm 1 are much smaller. The following lemma shows that asymptotically we use $\frac{1}{3}$ fewer nodes than in section 5.

LEMMA 6.3. *Algorithm* 1 *returns representation tree G for the linearization of a polynomial $p(x,y)$ of degree n of size*

$$\theta(n) = |G| = \begin{cases} \frac{1}{6}n(n+5), & n = 3k \text{ or } n = 3k+1, \\ \frac{1}{6}n(n+5) - \frac{1}{3}, & n = 3k+2. \end{cases} \tag{6.2}$$

Proof. It follows from the recursion in the algorithm (see Figure 7) that the number of nodes satisfies the recurrence equation

$$\theta(n) = n + 1 + \theta(n-3).$$

The solution of this equation with the initial values $\theta(1) = 1$, $\theta(2) = 2$, and $\theta(3) = 4$ is (6.2). □

For generic polynomials of degrees $n = 3$ and $n = 4$ it turns out to be possible to modify the construction and save one node in the representation tree. The main idea is to apply a linear substitution of variables x and y in the preliminary phase to the polynomial $p(x, y)$ to eliminate some of the terms. This implies that the resulting matrices are of orders 3 ($n = 3$) and 5 ($n = 4$), instead of orders 4 and 6 as seen before, and it also reduces the size of the matrices for $n = 3k$ and $n = 3k + 1$ by 1. We give details in the following two subsections.

6.1. The special case $n = 3$. We consider a cubic bivariate polynomial $p(x, y) = \alpha_{00} + \alpha_{10}x + \alpha_{01}y + \cdots + \alpha_{30}x^3 + \cdots + \alpha_{03}y^3$, where we can assume that $\alpha_{30} \neq 0$. We introduce a linear substitution of the form $x = \widetilde{x} + s\widetilde{y} + t$ and $y = \widetilde{y}$, where s is such that

$$h(s) := \alpha_{30}s^3 + \alpha_{21}s^2 + \alpha_{12}s + \alpha_{03} = 0 \tag{6.3}$$

and $t = -\frac{\alpha_{20}s^2 + \alpha_{11}s + \alpha_{02}}{h'(s)}$.

The substitution is well defined if s is a single root of (6.3) and the only situation where this is not possible is when h has a triple root. This substitution transforms $p(x, y)$ into a polynomial $\widetilde{p}(\widetilde{x}, \widetilde{y})$ such that its coefficients $\widetilde{\alpha}_{03}$ and $\widetilde{\alpha}_{02}$ are both zero. If we apply Algorithm 1 to $\widetilde{p}(\widetilde{x}, \widetilde{y})$ and choose $\zeta_1 = 0$ for the first zero, then the remainder in step 4 is zero and we get 3×3 matrices $\widetilde{A}$, $\widetilde{B}$, and $\widetilde{C}$ such that $\det(\widetilde{A} + \widetilde{x}\widetilde{B} + \widetilde{y}\widetilde{C}) = \widetilde{p}(\widetilde{x}, \widetilde{y})$. Now, it is easy to see that for $A = \widetilde{A} - t\widetilde{B}$, $B = \widetilde{B}$, and $C = \widetilde{C} - s\widetilde{B}$, $\det(A + xB + yC) = p(x, y)$.

Example 6.4. We take the recurrent example $p(x, y) = 1 + 2x + 3y + 4x^2 + 5xy + 6y^2 + 7x^3 + 8x^2y + 9xy^2 + 10y^3$ (see Examples 5.3 and 6.2). If we take $s = 1.1269$ (see (6.3)) and $t = -0.30873$, then substitution $x = \widetilde{x} + s\widetilde{y} + t$ and $y = \widetilde{y}$ changes $p(x, y)$ into a polynomial

$$\begin{aligned} p(\widetilde{x}, \widetilde{y}) = {} & 0.55782 + 1.5317\widetilde{x} + 0.49276\widetilde{y} - 2.4833\widetilde{x}^2 + 5.6571\widetilde{x}\widetilde{y} + 7\widetilde{x}^3 \\ & - 15.665\widetilde{x}^2\widetilde{y} + 17.637\widetilde{x}\widetilde{y}^2. \end{aligned}$$

Algorithm 1 gives 3×3 matrices $\widetilde{A}$, $\widetilde{B}$, and $\widetilde{C}$ such that $\det(\widetilde{A} + \widetilde{x}\widetilde{B} + \widetilde{y}\widetilde{C}) = \widetilde{p}(\widetilde{x}, \widetilde{y})$, from which matrices

$$A = \begin{bmatrix} 1.0307 & -0.76665 & 2.1611 \\ -0.30873 & 1 & 0 \\ 0 & -0.30873 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1.5317 & -2.4833 & 7 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \quad \text{and}$$

$$C = \begin{bmatrix} 2.2189 & 2.8587 & 0.00559 + 7.8813i \\ -1.1269 & 0 & 0 \\ 0 & -0.0079857 + 1.1259i & 0 \end{bmatrix},$$

such that $\det(A + xB + yC) = p(x, y)$, are obtained and we have a 3×3 linearization.

6.2. The special case $n = 4$. Before we give a construction for a generic quartic bivariate polynomial, we consider a particular case when a polynomial $p(x,y) = \sum_{j=0}^{4}\sum_{k=0}^{4-j}\alpha_{jk}x^jy^k$ of degree 4 is such that $\alpha_{30} = \alpha_{40} = \alpha_{03} = \alpha_{04} = 0$. In this case five nodes are enough to represent the polynomial $p(x,y)$. The representation tree for the polynomial $p(x,y)$ is presented in Figure 9, where ζ_1 and ζ_2 are the zeros of $\alpha_{31}\zeta^2 + \alpha_{22}\zeta + \alpha_{13}$.

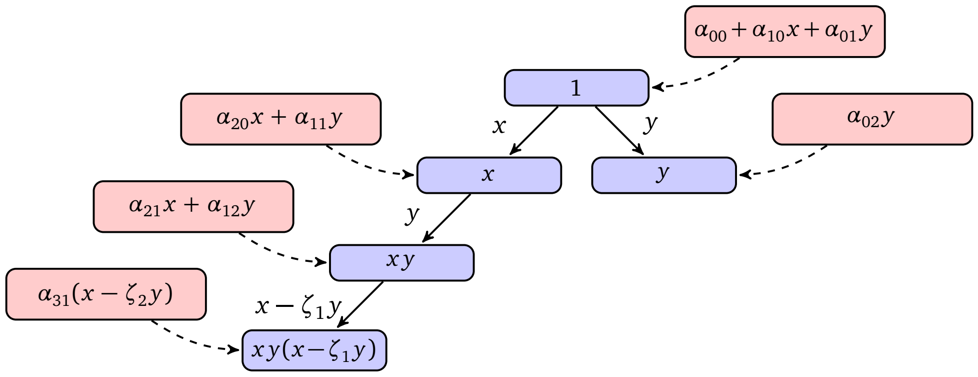
$\alpha_{00}+\alpha_{10}x+\alpha_{01}y$
1
x
y
$\alpha_{20}x+\alpha_{11}y$
$\alpha_{02}y$
x
y
y
$\alpha_{21}x+\alpha_{12}y$
xy
$\alpha_{31}(x-\zeta_2 y)$
$x-\zeta_1 y$
$xy(x-\zeta_1 y)$

FIG. 9. *A representation tree and a linearization for a polynomial* $p(x,y) = \sum_{j=0}^{4}\sum_{k=0}^{4-j}\alpha_{jk}x^jy^k$ *of degree* 4 *such that* $\alpha_{30} = \alpha_{40} = \alpha_{03} = \alpha_{04} = 0$.

For a generic quartic polynomial we first transform it into one with zero coefficients at x^3, x^4, y^3, and y^4. Except for very special polynomials, we can do this with a combination of two linear substitutions. Similarly as in case $n = 3$, we first introduce a linear substitution of the form $x = \widetilde{x} + s\widetilde{y} + t$ and $y = \widetilde{y}$, where s is such that

$$h(s) := \alpha_{40}s^4 + \alpha_{31}s^3 + \alpha_{22}s^2 + \alpha_{13}s + \alpha_{04} = 0$$

and $t = -\frac{\alpha_{30}s^3+\alpha_{21}s^2+\alpha_{12}s+\alpha_{03}}{h'(s)}$.

The substitution is well defined if s is a single root of $h(s)$. After the substitution we have a polynomial $\widetilde{p}(\widetilde{x},\widetilde{y})$ such that its coefficients $\widetilde{\alpha}_{04}$ and $\widetilde{\alpha}_{03}$ are both zero. On this polynomial we apply a new substitution $\widetilde{x} = \widehat{x}$ and $\widetilde{y} = u\widehat{x} + \widehat{y} + v$, where

$$g(u) := \widetilde{\alpha}_{40} + \widetilde{\alpha}_{31}u + \widetilde{\alpha}_{22}u^2 + \widetilde{\alpha}_{13}u^3 = 0$$

and $v = -\frac{\widetilde{\alpha}_{30}+\widetilde{\alpha}_{21}u+\widetilde{\alpha}_{12}u^2}{g'(u)}$.

This substitution is well defined if u is a single root of $g(u)$; therefore, both substitutions exist for a generic polynomial of degree 4.

After the second substitution we get a polynomial $\widehat{p}(\widehat{x},\widehat{y})$ such that its coefficients $\widehat{\alpha}_{30}$, $\widehat{\alpha}_{40}$, $\widehat{\alpha}_{03}$, and $\widehat{\alpha}_{04}$ are all zero. For such a polynomial we can construct a representation with matrices 5×5 as presented in Figure 9. This gives 5×5 matrices $\widehat{A}$, $\widehat{B}$, and $\widehat{C}$ such that $\det(\widehat{A} + \widehat{x}\widehat{B} + \widehat{y}\widehat{C}) = \widehat{p}(\widehat{x},\widehat{y})$. Finally, if we take

$$A = \widehat{A} - t\widehat{B} - (v - tu)\,\widehat{C}, \quad B = \widehat{B} - u\,\widehat{C}, \quad C = \widehat{C} - s\,\widehat{B},$$

then $\det(A + x\,B + y\,C) = p(x,y)$.

If we add the constructions from subsections 6.1 and 6.2 as special cases to Algorithm 1, then we save one node for all generic polynomials of degree $n = 3k$ or $n = 3k+1$. Although this advantage seems to be modest, numerical results in the following section point out that for small n this does speed up the computation of the zeros considerably (for instance, for $n = 6$ the corresponding Δ-matrices are of order $10^2 = 100$ instead of $11^2 = 121$).

7. Numerical examples.

Determinantal representations from sections 5 and 6 can be used to numerically solve a system of two bivariate polynomials. We first linearize the problem as a two-parameter eigenvalue problem and then solve it with the method for singular two-parameter eigenvalue problems from [33], which is implemented in [39]. Subsequently, we refine the solutions by two steps of Newton's method. We refer to the numerical methods that use linearizations from sections 5 and 6 as `Lin1` and `Lin2`, respectively. In the first example we take polynomials with random coefficients, while the second example considers some challenging benchmark polynomials.

The following numerical examples were obtained on a 64-bit Windows version of MATLAB R2015a running on an Intel Core i5-4670 3.40 GHz processor with 16 GB of RAM.

Example 7.1. We compare `Lin1` and `Lin2` to the following methods:

(a) `NSolve` in Mathematica 9 [51],

(b) `PHCLab` 1.04 [12] running `PHCpack` 2.3.84,

(c) `BertiniLab` 1.4 [37] running `Bertini` 1.5 [3],

(d) `NAClab` 3.0, a MATLAB toolbox for numerical algebraic computation [52],

where the last three methods use homotopy. We compare methods on systems of full bivariate polynomials of the same degree, whose coefficients are random real numbers uniformly distributed on $[0, 1]$ or random complex numbers, such that real and imaginary parts are both uniformly distributed on $[0, 1]$. We also tested `NSolve` in Mathematica 10.1, but we do not report the results, as it is slower than Mathematica 9 for polynomials of small degree.

The results are presented in Table 1. For each n we run all methods on the same set of 50 random polynomial systems and measure the average time. `Lin1` and Mathematica's `NSolve` work faster for polynomials with real coefficients, while this does not make a difference for `Lin2`, `PHCLab`, `BertiniLab`, and `NAClab`. Therefore, the results in the table for `Lin2` and `PHCLab` are an average of 50 real and 50 complex examples. Clearly, if `Lin1` is applied to a polynomial with real coefficients, then matrices Δ_0, Δ_1, and Δ_2 are real. If we apply `Lin2`, then the matrices are complex in general as roots of univariate polynomials are used in the construction. Although the complex arithmetic is more expensive than the real one, complex eigenproblems from `Lin2` are so small that they are solved faster than the larger real problems from `Lin1`. The sizes of Δ-matrices obtained in `Lin1` and `Lin2` are presented in Table 2.

Computational times for all methods except `NSolve` are very similar for each of the 50 test problems of the same degree, with the small exception that `Lin1` and `Lin2` failed to compute solutions in 5 and 6 of the 800 systems, respectively, and were therefore successfully restarted with polynomials where variables x and y are interchanged. On the other hand, `NSolve` needs substantially more time for certain systems. For example, for complex polynomials of degree $n = 7$, `NSolve` needed approximately $0.65\,s$ for 44 of the 50 examples and $3.8\,s$ for the additional 6 examples. That explains why the average time for `NSolve` ($\mathbb{C}$) in case $n = 7$ is just slightly smaller than in case $n = 8$. `Lin1` is competitive in particular for real systems of degree $n \leq 7$, while `Lin2` is the fastest method for real or complex systems of degree $n \leq 9$. For $n \geq 10$ `PHCLab` becomes the fastest method.

Table 1
Average computational times in milliseconds for `Lin1`, `Lin2`, `NSolve`, *and* `PHCLab` *for random full bivariate polynomial systems of degree* 3 *to* 10. *For* `Lin1` *and* `NSolve` *results are separated for real* ($\mathbb{R}$) *and complex polynomials* ($\mathbb{C}$). *Notice that these are the running times; the accuracy of the methods varies, as we discuss in the text.*

n	`Lin1` ($\mathbb{R}$)	`Lin1` ($\mathbb{C}$)	`Lin2`	`PHCLab`	`NSolve`($\mathbb{C}$)	`NSolve`($\mathbb{R}$)	`BertiniLab`	`NAClab`
3	5	5	4	109	110	19	247	112
4	9	9	8	122	184	35	331	188
5	16	20	12	140	292	68	478	297
6	36	52	24	165	447	111	677	434
7	73	131	56	198	1043	336	971	605
8	192	356	122	244	1088	352	1585	844
9	461	1090	271	309	1815	833	2510	1110
10	1433	3411	631	383	4403	1968	4246	1432

Table 2
Size of Δ*-matrices for* `Lin1` *and* `Lin2`.

Method	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$	$n=10$
`Lin1`	25	64	121	225	361	576	841	1225
`Lin2`	9	25	64	100	169	289	400	576

Beside the computational time, accuracy and reliability are other important factors. The only methods that return all solutions in all examples are `Lin1`, `Lin2`, and `NSolve`, but the results of `NSolve` are on average much less accurate than the results of all other methods. As a measure of accuracy we use the maximum value of

$$\max(|p_1(x_0,y_0)|,|p_2(x_0,y_0)|)\ \|J^{-1}(x_0,y_0)\|, \tag{7.1}$$

where $J(x_0,y_0)$ is the Jacobian matrix of p_1 and p_2 at (x_0,y_0), over all computed zeros (x_0,y_0). $\|J^{-1}(x_0,y_0)\|$ is an absolute condition number of a zero (x_0,y_0) and we assume that in random examples all zeros are simple. For a good method (7.1) should be as small as possible.

For degree $n \le 9$, `Lin2` is the fastest method and usually also a very accurate one. It is never significantly less accurate than the others, so it clearly wins in this case. The methods that are based on the homotopy sometimes fail to compute all the solutions. This happens to `BertiniLab`, `PHCLab`, and `NAClab` in 1, 5, and 54 of 800 systems. As the methods are using random initial systems, a possible remedy is to run them again.

We remark that every one fewer node in the representation tree really makes a difference. For instance, if we do not apply the special case for $n=4$ in subsection 6.2, then the Δ-matrices for `Lin2` for polynomial systems of degree $n=10$ are of size 625×625 instead of 576×576 and the average computational time rises from $0.63\,s$ to $0.72\,s$.

Example 7.2. We test `Lin1` and `Lin2` on 25 examples `ex001` to `ex025` from [6]. This set contains challenging benchmark problems with polynomials of small degree from $(3,2)$ to $(11,10)$ that have many multiple zeros and usually have fewer solutions than a generic pair of the same degrees. `Lin1` and `Lin2` performed satisfactorily on most examples, but, using default parameters they also failed on some. Instead of giving the details for all 25 examples, we give the key observations.

- Multiple zeros can present a problem for the algorithm from [17] that is used to solve the projected regular problem $\widetilde{\Delta}_1\, w = x\,\widetilde{\Delta}_0\, w$, $\widetilde{\Delta}_2\, w = y\,\widetilde{\Delta}_0\, w$ that is obtained from (4.2) by the modified staircase algorithm from [33]. The

QZ algorithm is first applied to $\widetilde{\Delta}_1 w = x \widetilde{\Delta}_0 w$ and then $\widetilde{\Delta}_2 w = y \widetilde{\Delta}_0 w$ is multiplied by Q and Z. The eigenvalues are clustered along the diagonal so that multiple eigenvalues x should be in the same block. For several of the 25 examples with eigenvalues of high multiplicity the clustering criteria have to be adapted; otherwise the results are not very accurate. An example is `ex004`.

- `Lin2` is faster, but the accuracy can be lost and the method can fail if the polynomial in step 1 of Algorithm 1 has multiple zeros; an example is p_2 from `ex008`. In such a case it helps to compute the roots with a solver `uvFactor` from NAClab that can compute multiple roots of a polynomial more accurately.
- We get very good results in example `ex005` with the system $x^9+y^9-1=0$ and $x^{10}+y^{10}-1=0$ using `Lin2`. In this case `Lin2` returns optimal determinantal representations with matrices of size 9×9 and 10×10, respectively. The obtained two-parameter eigenvalue problem is not singular and we get the solutions in $0.03\,s$, while `PHCLab` and `NSolve` need $0.25\,s$ and $0.29\,s$, respectively. For comparison, `Lin1`, applied to the same problem, returns Δ-matrices of size 1015×1015, while `Lin2` gives Δ-matrices of size 90×90.
- `Lin1` is slower but can be more accurate. Because there is no computation in the construction, no errors are introduced in the construction of the linearization.
- For some examples `Lin1` and `Lin2` fail to compute the solutions using default parameters. In all such cases we can adapt the parameters so that the methods compute the right number of solution accurately. In some cases, when `Lin1` and `Lin2` fail, the solution is to simply exchange the variables x and y or to apply a linear substitution $x \to \alpha\widetilde{x} + \beta\widetilde{y}$, $y \to \gamma\widetilde{x} + \delta\widetilde{y}$ to the original polynomials. Such examples are `ex016` and `ex018`.
- `Lin2` is the fastest method for all examples except for `ex014`, where `PHCLab` is faster. The polynomials in this example are of degrees 11 and 10. `NSolve` always finds all solutions but is slower than `Lin1` and `Lin2`. `PHCLab` usually does not find all instances of multiple eigenvalues and thus returns many fewer zeros.

Example 7.3. Encouraged by the good results for `ex005` in Example 7.2, we carry out some experiments with polynomials of form $p(x,y) = \alpha_{n0}x^n + \cdots + \alpha_{0n}y^n + h(x,y)$, where $h(x,y)$ is a polynomial of small degree $m \ll n$. For such polynomials Algorithm 1 returns matrices of size $n+1+\theta(m)$ or even smaller. For example, it is easy to see that for $m=1$ we get linearization of the smallest possible size $n\times n$. We compared `Lin2`, `PHCLab`, and `NSolve`. Computational times for random polynomials with complex coefficients of the above form are presented in Table 3. As n increases, `PHCLab` becomes faster then `Lin2`, but for smaller degree `Lin2` might be the preferred method.

8. Conclusions.

We have proposed two linearizations for bivariate polynomials. The first linearization does not involve any computation as the coefficients of the polynomials appear as (block) coefficients of the matrices A, B, and C. This linearization is suitable for both scalar and matrix bivariate polynomials. The second linearization, useful for scalar polynomials, involves little computation and returns much smaller matrices. They are still larger than the theoretically smallest possible size $n\times n$, but their construction is very simple and fast. Moreover, while the asymptotic order is $\frac{1}{6}n^2$, the order for small n is about $2n$; for polynomials of degrees 3 and 4 we have

Table 3
Computational times in seconds for `Lin2`*,* `PHCLab`*, and* `NSolve` *for systems of two polynomials of the form* $p(x,y)=\alpha_{n0}x^n+\cdots+\alpha_{0n}y^n+h(x,y)$*, where degree of* $h(x,y)$ *is* $m \ll n$*.*

n	m	`Lin2`	`PHCLab`	`NSolve`
15	1	0.16	0.46	1.58
15	3	0.20	0.55	1.77
20	1	0.65	1.00	4.24
20	3	0.75	1.06	4.59
25	1	2.51	2.07	9.76
25	3	2.47	1.95	10.30

presented determinantal representations of orders 3 and 5, respectively.

As an application we have presented a method for finding roots of two bivariate polynomials. We have shown that the presented approach, where the polynomial system is first linearized into a two-parameter eigenvalue problem, which is later solved by a modified staircase method, is numerically feasible and gives very promising results for polynomials of degree $n \lesssim 10$, as well as for polynomials of higher degree but with few terms. Any further results on even smaller determinantal representations that can be efficiently constructed numerically could enlarge the above degree.

Acknowledgments. The first author wishes to thank the NWO for the visitor's grant and the CASA group for the hospitality. We thank the referees for their useful suggestions.

REFERENCES

[1] F. V. Atkinson, *Multiparameter Eigenvalue Problems*, Academic Press, New York, 1972.

[2] C. Bajaj, T. Garrity, and J. Warren, *On the Applications of Multi-Equational Resultants*, Technical report, Purdue University, 1988.

[3] D. J. Bates, J. H. Husenstein, A. J. Sommese, and C. W. Wampler, *Bertini: Software for Numerical Algebraic Geometry*, bertini.nd.edu; doi: dx.doi.org/10.7274/R0H41PB5.

[4] P. Berman, A. Bhattacharyya, K. Makarychev, S. Raskhodnikova, and G. Yaroslavtsev, *Approximation algorithms for spanner problems and directed Steiner forest*, Inform. Comput., 222 (2013), pp. 93–107.

[5] S. Beyme and C. Leung, *A stochastic process model of the hop count distribution in wireless sensor networks*, Ad Hoc Networks, 17 (2014), pp. 60–70.

[6] L. Busé, H. Khalil, and B. Mourrain, *Resultant-based methods for plane curves intersection problems*, in Proceedings of the 8th International Conference on Computer Algebra in Scientific Computing, Lecture Notes in Comput. Sci. 3718, Springer-Verlag, Berlin, 2005, pp. 75–92.

[7] R. H. Byrd, R. B. Schnabel, and G. A. Schultz, *Approximate solution of the trust region problem by minimization over two-dimensional subspaces*, Math. Program., 40, 1 (1988), pp. 247–263.

[8] M. Costa, V. Koivunen, and A. Richter, *Low complexity azimuth and elevation estimation for arbitrary array configurations*, in Proceedings of Acoustics, Speech and Signal Processing, IEEE, 2009, pp. 2185–2188.

[9] L. E. Dickson, *Determination of all general homogeneous polynomials expressible as determinants with linear elements*, Trans. Amer. Math. Soc., 22 (1921), pp. 167–179.

[10] A. Dixon, *Note on the reduction of a ternary quartic to a symmetrical determinant*, Proc. Camb. Phil. Soc., 11 (1902), pp. 350–351.

[11] P. Dreesen, K. Batselier, and B. De Moor, *Back to the roots: polynomial system solving, linear algebra, systems theory*, in Proceedings of the 16th IFAC Symposium on System Identification, Brussels, Belgium, 2012 pp. 1203–1208.

[12] Y. Guan and J. Verschelde, *PHClab: A MATLAB/Octave Interface to PHCpack*, in Software for Algebraic Geometry, M. Stillman, J. Verschelde, and N. Takayama, eds, IMA Vol. Math. Appl. 148, Springer, New York, 2008, pp. 15–32.

[13] B. Hanzon and J. Maciejowski, *Constructive algebra methods for the L2-problem for stable linear systems*, Automatica, 32 (1996), pp. 1645–1657.

[14] G. F. Hatke, *Superresolution source location with planar arrays*, Lincoln Lab. J., 10 (1997).

[15] J. W. Helton, S. A. McCullough, and V. Vinnikov, *Noncommutative convexity arises from linear matrix inequalities*, J. Funct. Anal., 240 (2006), pp. 105–191.

[16] J. W. Helton and V. Vinnikov, *Linear matrix inequality representation of sets*, Comm. Pure Appl. Math., 60 (2007), pp. 654–674.

[17] M. E. Hochstenbach, T. Košir, and B. Plestenjak, *A Jacobi–Davidson type method for the nonsingular two-parameter eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 477–497.

[18] M. E. Hochstenbach, A. Muhič, and B. Plestenjak, *On linearizations of the quadratic two-parameter eigenvalue problems*, Linear Algebra Appl., 436 (2012), pp. 2725–2743.

[19] M. E. Hochstenbach, A. Muhič, and B. Plestenjak, *A Jacobi–Davidson method for polynomial two–parameter eigenvalue problems*, J. Comput. Appl. Math., 288 (2015), pp. 251–263.

[20] M. E. Hochstenbach, H. A. van der Vorst, *Alternatives to the Rayleigh quotient for the quadratic eigenvalue problem*, SIAM J. Sci. Comput., 25 (2003), pp. 591–603.

[21] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[22] E. Jarlebring and M. E. Hochstenbach, *Polynomial two-parameter eigenvalue problems and matrix pencil methods for stability of delay-differential equations*, Linear Algebra Appl., 431 (2009), pp. 369–380.

[23] G. Jónsson and S. Vavasis, *Accurate solution of polynomial equations using Macaulay resultant matrices*, Math. Comp., 74 (2005), pp. 221–262.

[24] D. Jungnickel, *Graphs, Networks and Algorithms*, 4th ed., Springer, Heidelberg, 2013.

[25] V. B. Khazanov, *To solving spectral problems for multiparameter polynomial matrices*, J. Math. Sci., 141 (2007), pp. 1690–1700.

[26] K. H. Ko, T. Sakkalis, and N. M. Patrikalakis, *Nonlinear polynomial systems: Multiple roots and their multiplicities*, in Proceedings of Shape Modeling Applications, IEEE, 2004, pp. 87–98.

[27] P. Lancaster and P. Psarrakos, *A Note on Weak and Strong Linearizations of Regular Matrix Polynomials*, NA Report 470, Manchester Centre for Computational Mathematics, 2005.

[28] J. B. Lasserre, M. Laurent, and P. Rostalski, *A unified approach to computing real and complex zeros of zero-dimensional ideals*, in *Emerging Applications of Algebraic Geometry*, M. Putinar and S. Sullivant, eds., IMA Vol. Math. Appl. 149, Springer-Verlag, New York, 2009, pp. 125–155.

[29] C. Lennerz and E. Schömer, *Efficient distance computation for quadratic curves and surfaces*, in Proceedings of Geometric Modeling and Processing, IEEE, 2002, pp. 60–69.

[30] *MATLAB*, The MathWorks, Natick, MA.

[31] A. Muhič, *Numerical Methods for Singular Multiparameter Eigenvalue Problems*, Ph.D. thesis, University of Ljubljana, 2011.

[32] A. Muhič and B. Plestenjak, *On the singular two-parameter eigenvalue problem*, Electron. J. Linear Algebra, 18 (2009), pp. 420–437.

[33] A. Muhič and B. Plestenjak, *On the quadratic two-parameter eigenvalue problem and its linearization*, Linear Algebra Appl., 432 (2010), pp. 2529–2542.

[34] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Computing the common zeros of two bivariate functions via Bézout resultants*, Numer. Math. (2014): pp. 1–29.

[35] *NCAlgebra: Non Commutative Algebra Packages*, math.ucsd.edu/~ncalg.

[36] T. Netzer and A. Thom, *Polynomials with and without determinantal representations*, Linear Algebra Appl., 437 (2012), pp. 1579–1595.

[37] A. Newell, *BertiniLab: Toolbox for solving polynomial systems*, MATLAB Central File Exchange, www.mathworks.com/matlabcentral/fileexchange/48536-bertinilab (2015).

[38] D. Plaumann, R. Sinn, D. E. Speyer, and C. Vinzant, *Computing Hermitian Determinantal Representations of Hyperbolic Curves*, arXiv:1504.06023[math.AG], 2015.

[39] B. Plestenjak, MultiParEig: *Toolbox for multiparameter eigenvalue problems*, MATLAB Central File Exchange, www.mathworks.com/matlabcentral/fileexchange/47844-multipareig, (2015).

[40] R. Quarez, *Symmetric determinantal representation of polynomials*, Linear Algebra Appl., 436 (2012), pp. 3642–3660.

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

[42] L. Sorber, *Data Fusion—Tensor Factorizations by Complex Optimization*, Ph.D. thesis, KU Leuven, 2014.

[43] 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.

[44] H. J. Stetter, *Numerical Polynomial Algebra*, SIAM, Philadelphia, 2004.

[45] B. Sturmfels, *Solving Systems of Polynomial Equations*, CBMS Reg. Conf. Ser. Math. 97, AMS, Providence, RI, 2002.

[46] A. Townsend and L. N. Trefethen, *An extension of Chebfun to two dimensions*, SIAM J. Sci. Comput., 35 (2013), pp. 495–518.

[47] J. Van der Laar, *Mimo Instantaneous Blind Identification and Separation Based on Arbitrary Order Temporal Structure in the Data*, Ph.D. thesis, TU Eindhoven, 2007.

[48] P. van Dooren, *The computation of Kronecker's canonical form of a singular pencil*, Linear Algebra Appl., 27 (1979), pp. 103–141.

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

[50] V. Vinnikov, *LMI representations of convex semialgebraic sets and determinantal representations of algebraic ypersurfaces: Past, present, and future*, in *Mathematical Methods in Systems, Optimization, and Control: Festschrift in Honor of J. William Helton*, H. Dym. M. C. de Oliveira, and M. Putinar, eds., Oper. Theory Adv. Appl. 222, Birkhäuser, 2012 pp. 325–349.

[51] *Mathematica, Version* 9.0, Wolfram Research, Inc., Champaign, IL, 2012.

[52] Z. Zeng and T.-Y. Li, *NAClab: A Matlab toolbox for numerical algebraic computation*, ACM Commun. Comput. Algebra, 47 (2013), pp. 170–173.

[53] K. Zhou and S. I. Roumeliotis, *Multirobot active target tracking with combinations of relative observations*, IEEE J. Robot. Automat., 27 (2011), pp. 678–695.