NUMERICAL SOLUTION OF BIVARIATE AND POLYANALYTIC POLYNOMIAL SYSTEMS

LAURENT SORBER[†§], MARC VAN BAREL[†], AND LIEVEN DE LATHAUWER[‡§]

Abstract. Finding the real solutions of a bivariate polynomial system is a central problem in robotics, computer modeling and graphics, computational geometry and numerical optimization. We propose an efficient and numerically robust algorithm for solving bivariate and polyanalytic polynomial systems using a single generalized eigenvalue decomposition. In contrast to existing eigen-based solvers, the proposed algorithm does not depend on Gröbner bases or normal sets, nor does it require computing eigenvectors or solving additional eigenproblems to recover the solution. The method transforms bivariate systems into polyanalytic systems and then uses resultants in a novel way to project the variables onto the real plane associated with the two variables. Solutions are returned counting multiplicity and their accuracy is maximized by means of numerical balancing and Newton–Raphson refinement. Numerical experiments show that the proposed algorithm consistently recovers a higher percentage of solutions and is at the same time significantly faster and more accurate.

1. Introduction. Computing the roots of a set of polynomial equations is a well studied problem for which a wide range of techniques have been developed. These can roughly be categorized into numeric, symbolic and hybrid methods. Numeric techniques based on Newton's method [35] are useful for finding solutions locally. However, there is no guarantee of finding all solutions and multiple roots may cause convergence issues. Methods based on interval arithmetic [32] are robust, but can suffer from slow convergence and require bounds on the solutions of interest. In the context of optimization, there also exist various convex relaxation methods for global minimization of polynomials such as a semidefinite programming relaxation [23,36,42], but these only return a lower bound on the objective function in general. Gröbner bases [7, 15] and resultants [14] are symbolic methods that originate from algebraic geometry. They are both used to eliminate variables, essentially reducing the problem to finding the roots of univariate polynomials. However, Gröbner bases are expensive to compute and are often numerically ill-conditioned. On the other hand, resultants can introduce spurious solutions and have historically not led to numerically robust algorithms either. Homotopy continuation methods [26,50] are hybrid techniques that combine homotopy methods with numerical continuation methods. The idea is to start from a system with a known solution set, and then to numerically track the solutions

*Received by the editors DATE; accepted for publication (in revised form) DATE; published electronically DATE. The scientific responsibility rests with the authors. http://www.siam.org/journals/siopt/x-x/xxxxx.html

[†]Department of Computer Science, KU Leuven, Celestijnenlaan 200A, BE-3001 Leuven, Belgium (Laurent.Sorber@cs.kuleuven.be, Marc.VanBarel@cs.kuleuven.be). The first author is supported by a doctoral fellowship of the Flanders agency for Innovation by Science and Technology (IWT). The second author's research is supported by (1) the Research Council KU Leuven: (a) project OT/10/038, (b) CoE EF/05/006 Optimization in Engineering (OPTEC), and by (2) the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

[‡]Group Science, Engineering and Technology, KU Leuven Kulak, E. Sabbelaan 53, BE-8500 Kortrijk, Belgium (Lieven.DeLathauwer@kuleuven-kulak.be).

[§]STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10, BE-3001 Leuven, Belgium and iMinds Future Health Department (Lieven.DeLathauwer@esat.kuleuven.be). The third author's research is supported by (1) the Research Council KU Leuven: (a) GOA-MaNet, (b) CoE EF/05/006 Optimization in Engineering (OPTEC), (c) CIF1, (d) STRT1/08/023, (2) the Research Foundation Flanders (FWO): project G.0427.10N and by (3) the Belgian Network DYSCO: IUAP P7/19.

as the system is gradually transformed into the target system. Although homotopy continuation has been vastly improved in recent years, issues such as path crossing can decrease their robustness [33]. This problem is exacerbated in the bivariate case, where path crossing is more likely to happen than for systems of three or more variables.

In this article, we propose an efficient and numerically robust algorithm to compute the isolated real solutions of bivariate polynomial systems. Instead of using resultants to project the solution onto the complex plane associated with one of the two variables, we apply them in a novel way to project solutions onto the real plane associated with both variables. The key to this projection is to first transform the bivariate polynomial system into a so-called polyanalytic polynomial system. The projected system is then reduced to a polynomial eigenvalue problem, which can in turn be linearized into a generalized eigenvalue problem. As such, our method can be seen as a hybrid method, although symbolic manipulations are absent. In contrast to existing eigen-based solvers [3, 8, 11, 19, 24, 27, 28, 31, 46], the proposed algorithm does not depend on Gröbner bases or normal sets, nor does it require computing eigenvectors or solving additional eigenproblems to recover the solution. Moreover, the method returns solutions counting multiplicity and is also applicable to polyanalytic polynomial systems. We aim to maximize accuracy with numerical balancing and Newton–Raphson refinement, and present a robust method to filter out spurious solutions introduced by the resultant. In our numerical experiments, we compare performance with Chebfun2 [47], PHCpack [49], the RealSolving C library [38, 39], the RegularChains package [2] and Maple's BivatiatePolynomial method on a range of bivariate systems of varying difficulty. The resulting performance profiles show that the proposed algorithm consistently recovers a higher percentage of solutions and is at the same time significantly faster and more accurate.

The paper is organized as follows. In the next subsection we review our notation and introduce some basic definitions. In Section 2 we examine a classical approach to solving bivariate polynomial systems and propose an algorithm for solving both bivariate and polyanalytic polynomial systems with a single generalized eigenvalue decomposition. Furthermore, we also present several balancing steps and a robust strategy for removing spurious roots. In Section 3 we evaluate the performance of the proposed algorithm on a sizable set of low- to moderate-degree bivariate polynomial systems using performance profiles [13]. We conclude the paper in Section 4.

1.1. Notation and preliminaries. Vectors are denoted by boldface letters and are lower case, e.g., $\boldsymbol{a}$. Matrices are usually denoted by capital letters, e.g., A. Higher-order tensors are denoted by Euler script letters, e.g., $\mathcal{A}$. An entry of a vector $\boldsymbol{a}$, matrix A or tensor $\mathcal{A}$ is denoted by a_i, a_{ij} or $a_{ijk\ldots}$, depending on the number of modes. A colon is used to select all entries of a mode. For instance, $\boldsymbol{a}_{:j}$ corresponds to the jth column of a matrix A. When there is no confusion, we also use $\boldsymbol{a}_j$ to denote the jth column of the matrix A. Sequences are denoted by a superscript in parentheses, e.g., $\{A^{(n)}\}_{n=1}^{N}$. The superscripts $\cdot^{\mathrm{T}}$, $\cdot^{\mathrm{H}}$, $\cdot^{-1}$ and $\cdot^{\dagger}$ are used for the transpose, Hermitian conjugate, matrix inverse and Moore–Penrose pseudoinverse, respectively. The complex conjugate is denoted by an overbar, e.g., $\overline{a}$ is the complex conjugate of the scalar a. We use parentheses to denote the concatenation of two or more vectors, e.g., $(\boldsymbol{a}, \boldsymbol{b})$ is equivalent to $[\boldsymbol{a}^{\mathrm{T}} \quad \boldsymbol{b}^{\mathrm{T}}]^{\mathrm{T}}$. The $n \times n$ identity matrix is denoted by $\mathbb{I}_n$, and the all-zero and all-one $m \times n$ matrices by $0_{m\times n}$ and $1_{m\times n}$, respectively.

Let $\mathcal{B}$ be a basis for the ring of polynomials $\mathbb{C}[x]$ with basis polynomials $B_n(x)$ of degree n. For example, if $\mathcal{B}$ is the monomial basis or Chebyshev basis, the corresponding basis polynomials $B_n(x)$ are x^n and $T_n(x) := \cos(n \arccos(x))$, respectively. Given a choice of basis, we associate a coefficient vector $\boldsymbol{p} \in \mathbb{C}^{d_p+1}$, $p_{d_p+1} \neq 0$, with polynomials

$$p(x) := \begin{bmatrix} B_0(x) & B_1(x) & \cdots & B_{d_p}(x) \end{bmatrix} \cdot \boldsymbol{p}$$

of degree d_p. Analogously, we associate a coefficient matrix $P \in \mathbb{C}^{(d_p^{(y)}+1)\times(d_p^{(x)}+1)}$, $\max_i |p_{i,d_p^{(x)}+1}| \neq 0 \wedge \max_j |p_{d_p^{(y)}+1,j}| \neq 0$, with bivariate polynomials

$$p(x,y) := \begin{bmatrix} B_0(y) & B_1(y) & \cdots & B_{d_p^{(y)}}(y) \end{bmatrix} \cdot P \cdot \begin{bmatrix} B_0(x) & B_1(x) & \cdots & B_{d_p^{(x)}}(x) \end{bmatrix}^{\mathrm{T}}$$

of coordinate degree $(d_p^{(x)}, d_p^{(y)})$. The total degree d_p of the polynomial $p(x,y)$ is defined as the largest value $i+j$ for which $p_{i+1,j+1} \neq 0$. We represent bivariate polynomials in product bases, i.e., as a linear combination of products $B_i(y)B_j(x)$.

The choice of basis can depend on the application and in which region the roots of interest lie. For example, the product monomial basis is orthogonal with respect to the inner product over the two unit circles in the complex planes associated with x and y, while the product Chebyshev basis is orthogonal on the unit square associated with x and y.

Substituting x and y by z and its complex conjugate $\overline{z}$ produces so-called *polyanalytic* polynomials $p(z,\overline{z})$. The term polyanalytic refers to the fact that the function is not analytic in z, but is analytic in z and $\overline{z}$ as a whole. Furthermore, it is not hard to show that if $p(z,\overline{z})$ is real-valued for all $z \in \mathbb{C}$, then P is Hermitian.

2. Solving systems of bivariate and polyanalytic polynomials.

2.1. The polynomial eigenvalue problem. In robotics [54], computer modeling and graphics [17, 20], computational geometry [4] and numerical optimization [9, 43], a common task is to compute the isolated real roots of bivariate polynomial systems

$$p(x,y) = q(x,y) = 0 \tag{2.1}$$

or the isolated complex roots of polyanalytic polynomial systems

$$r(z,\overline{z}) = s(z,\overline{z}) = 0, \tag{2.2}$$

where the associated coefficient matrices P, Q, R and S may be real- or complex-valued. For example, such systems appear as line and plane search subproblems in tensor optimization [43], and more generally also in polynomial optimization. Unconstrained tensor optimization problems are of the form

$$\underset{\boldsymbol{z} \in \mathbb{C}^n}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{M}(\boldsymbol{z}) - \mathcal{T}\|_{\mathrm{F}}^2, \tag{2.3}$$

where $\mathcal{T} \in \mathbb{C}^{I_1 \times \cdots \times I_N}$ is a given tensor and $\mathcal{M}$ is usually a multilinear function of the variables $\boldsymbol{z}$. A function is said to be multilinear in its argument $\boldsymbol{z}$ if for all i it is linear in z_i when the remaining variables z_j $(j \neq i)$ are fixed. Among other applications, (2.3) can be used to compute tensor decompositions such as the canonical polyadic

decomposition [10,18] and low multilinear rank approximation [22,48]. Given a current iterate $\boldsymbol{z}_k$ and two descent directions $\Delta\boldsymbol{z}_1$ and $\Delta\boldsymbol{z}_2$,

$$\underset{\alpha\in\mathbb{C}}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{M}(\boldsymbol{z}_k+\alpha\Delta\boldsymbol{z}_1)-\mathcal{T}\|_{\text{F}}^2 \tag{2.4a}$$

$$\underset{\alpha,\beta\in\mathbb{R}}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{M}(\boldsymbol{z}_k+\alpha\Delta\boldsymbol{z}_1+\beta\Delta\boldsymbol{z}_2)-\mathcal{T}\|_{\text{F}}^2 \tag{2.4b}$$

are referred to as complex line search and real plane search subproblems, respectively. Let $f_{\text{LS}}(\alpha,\overline{\alpha})$ and $f_{\text{PS}}(\alpha,\beta)$ be the objective functions of (2.4a) and (2.4b), respectively, then the global line and plane search minimizers are solutions of the bivariate and polyanalytic polynomial systems

$$\frac{\partial f_{\text{LS}}}{\partial\alpha}=\frac{\partial f_{\text{LS}}}{\partial\overline{\alpha}}=0 \qquad \text{and} \qquad \frac{\partial f_{\text{PS}}}{\partial\alpha}=\frac{\partial f_{\text{PS}}}{\partial\beta}=0,$$

where $\frac{\partial}{\partial\alpha}$ ($\frac{\partial}{\partial\overline{\alpha}}$) is a Wirtinger derivative and acts as partial derivative with respect to α ($\overline{\alpha}$), while treating $\overline{\alpha}$ (α) as constant [44].

Two-dimensional subspace minimization [9] is similar to the plane search (2.4b), where $\mathcal{M}$ is a linear approximation of a nonlinear residual function F and a norm constraint on the step size such as $\|\alpha\Delta\boldsymbol{z}_1+\beta\Delta\boldsymbol{z}_2\|\leq\delta$ is added. The stationary points of the resulting optimization problem are the solutions of a bivariate polynomial system in which the polynomials have total degree 2. In [43], it is shown that two-dimensional subspace minimization can be generalized to higher-order Taylor series expansions of the residual function F and that the resulting optimization problem is a bivariate polynomial system with polynomials of total degree 2 and $2d$, where d is the order of the Taylor series approximation.

A widely used technique to solve bivariate polynomial systems (2.1) is to eliminate one of the two variables by reducing the problem to finding the roots of a so-called resultant polynomial. Let $V_{\mathbb{C}}$ denote the set $\{(x,y)\in\mathbb{C}^2\,|\,p(x,y)=q(x,y)=0\}$. If p and q have no common factor, then by Bézout's theorem $V_{\mathbb{C}}$ is zero-dimensional and contains at most d_pd_q elements. The resultant of p and q is an algebraic description of the projections of the set $V_{\mathbb{C}}$ onto the complex plane associated with x or y, depending on which variable is eliminated. In other words, the resultant is a polynomial whose roots include the set

$$V_{\mathbb{C}}^{(x)}:=\{x\in\mathbb{C}\,|\,\exists y\in\mathbb{C}:p(x,y)=q(x,y)=0\} \text{ or} \tag{2.5a}$$

$$V_{\mathbb{C}}^{(y)}:=\{y\in\mathbb{C}\,|\,\exists x\in\mathbb{C}:p(x,y)=q(x,y)=0\}. \tag{2.5b}$$

From here on, we assume the variable y is eliminated so that the resultant is a polynomial in x. Of course, the role of x and y may be reversed in the subsequent discussion.

Resultants have many formulations, but are typically represented by (ratios of) determinants. The two most common resultants are those of Sylvester and Bézout, the former of which is a corollary of the following lemma.

LEMMA 2.1. *Let $p(x,y)$ and $q(x,y)$ be two nonconstant bivariate polynomials. If $p(x,y)$ and $q(x,y)$ have a common zero $(x^*,y^*)\in\mathbb{C}^2$, then there are nonzero polynomials $u(y)$ and $v(y)$ satisfying*

$$p(x^*,y)u(y)+q(x^*,y)v(y)\equiv 0, \tag{2.6}$$

where $d_u<d_q^{(y)}$ and $d_v<d_p^{(y)}$.

Proof. Write $p(x,y)$ and $q(x,y)$ as

$$p(x,y) = \sum_{i=0}^{d_p^{(y)}} p_i(x) B_i(y) \quad \text{and} \quad q(x,y) = \sum_{i=0}^{d_g^{(y)}} q_i(x) B_i(y),$$

where $p_i(x)$ ($q_i(x)$) is the polynomial in x associated with the coefficient vector $\boldsymbol{p}_{i+1,:}$ ($\boldsymbol{q}_{i+1,:}$). Since $p(x^*,y^*) = q(x^*,y^*) = 0$, $p(x^*,y)$ and $q(x^*,y)$ will have a common factor $k(y) := (y-y^*)^n$ for some positive integer n. We have that $\frac{p(x^*,y)}{k(y)} q(x^*,y) = \frac{q(x^*,y)}{k(y)} p(x^*,y)$. Set $u(y) := q(x^*,y)/k(y)$ and $v(y) := -p(x^*,y)/k(y)$. □

Equation (2.6) expresses the fact that if (x^*,y^*) satisfies $p(x^*,y^*) = q(x^*,y^*) = 0$, then there exist polynomials $u(y)$ and $v(y)$ such that all coefficients of the polynomial $p(x^*,y)u(y) + q(x^*,y)v(y)$ are zero. Writing p, q, u and v in the basis $\{B_i(y)\}_{i=0}^{n-1}$, where $n := d_p^{(y)} + d_q^{(y)}$, these conditions are equivalent to the system

$$S^{p,q}(x^*) \cdot \boldsymbol{r} = \boldsymbol{0}, \tag{2.7}$$

where $\boldsymbol{r} := (\boldsymbol{u}, \boldsymbol{v})$ and $S^{p,q}(x)$ is a polynomial matrix called the Sylvester matrix. In other words, Lemma 2.1 states that $S^{p,q}(x)$ must be singular when evaluated at the x-coordinate of a common root. The Sylvester resultant is now defined as $\operatorname{res}^{p,q}(x) := \det(S^{p,q}(x))$. Clearly, the resultant is a polynomial in x and vanishes if and only if the Sylvester matrix is singular. The structure of the Sylvester matrix depends on the choice of basis corresponding to the variable y. Let $f(x,y)$ be a polynomial in the basis $\{B_i(y)\}$ with polynomial coefficients $f_i(x)$. Define the infinite Toeplitz, Hankel and shifted Toeplitz matrices

$$L^f(x) := \begin{bmatrix} f_0(x) & \\ \vdots & \ddots \\ f_{d_f^{(y)}}(x) & \ddots \\ & \ddots \end{bmatrix},$$

$$M^f(x) := \begin{bmatrix} f_0(x) & \cdots & f_{d_f^{(y)}}(x) & 0 & \cdots \\ \vdots & \iddots & \iddots & & \\ f_{d_f^{(y)}}(x) & \iddots & & & \\ 0 & & & & \\ \vdots & & & & \end{bmatrix} \quad \text{and}$$

$$N^f(x) := \begin{bmatrix} 0 & 0 & \cdots & & \\ 0 & f_0(x) & \cdots & f_{d_f^{(y)}}(x) & \\ \vdots & & \ddots & \ddots & \ddots \end{bmatrix},$$

respectively. For the monomial basis we have $y^i y^j = y^{i+j}$ so that the Sylvester matrix is given by

$$S^{p,q}(x) = \begin{bmatrix} L^p_{1:n,1:d_q^{(y)}}(x) & L^q_{1:n,1:d_p^{(y)}}(x) \end{bmatrix}.$$

For the Chebyshev basis we have $2T_i(y)T_j(y) = T_{i+j}(y) + T_{|i-j|}(y)$ so that the Sylvester matrix is, up to a scale factor, equal to

$$S^{p,q}(x) = \begin{bmatrix} {L^p}^{\mathrm{T}}_{1:n,1:d_q^{(y)}}(x) + {M^p}^{\mathrm{T}}_{1:n,1:d_q^{(y)}}(x) + {N^p}^{\mathrm{T}}_{1:n,1:d_q^{(y)}}(x) \\ {L^q}^{\mathrm{T}}_{1:n,1:d_p^{(y)}}(x) + {M^q}^{\mathrm{T}}_{1:n,1:d_p^{(y)}}(x) + {N^q}^{\mathrm{T}}_{1:n,1:d_p^{(y)}}(x) \end{bmatrix}^{\mathrm{T}}.$$

Importantly, the polynomial $\mathrm{res}^{p,q}(x)$ is in a sense a minimal characterization of the x-coordinates of the roots of the system $p(x,y) = q(x,y) = 0$. In fact, its roots are exactly the projected roots $V_{\mathbb{C}}^{(x)}$ together with roots at infinity of the form (x^*, ∞).

THEOREM 2.2 (see, e.g., [5]). *If p and q are coprime, the distinct roots of* $\mathrm{res}^{p,q}(x)$ *are the roots of the bivariate system $p(x,y) = q(x,y) = 0$ projected onto the complex plane associated with x and the roots of the leading coefficient system $p_{d_p^{(y)}}(x) = q_{d_q^{(y)}}(x) = 0$. More precisely,*

$$\{x \in \mathbb{C} \,|\, \mathrm{res}^{p,q}(x) = 0\} \;=\; V_{\mathbb{C}}^{(x)} \cup \{x \in \mathbb{C} \,|\, p_{d_p^{(y)}}(x) = q_{d_q^{(y)}}(x) = 0\}.$$

The multiplicity of a root x^* of $\mathrm{res}^{p,q}(x)$ *is the sum of the multiplicities of all solutions of $p(x,y) = q(x,y) = 0$ with x-coordinate x^*.*

To compute the roots of the resultant, one possible approach is to explicitly compute the resultant $\mathrm{res}^{p,q}(x)$ as the determinant of a polynomial matrix [40], after which the roots can be obtained as the eigenvalues of its companion matrix. However, it is well known that a polynomial's roots can be very sensitive to small changes in the polynomial's coefficients [52]. For this reason, the accuracy required of the resultant's coefficients makes this approach prohibitively expensive. Instead, the resultant's roots can be obtained directly without constructing $\mathrm{res}^{p,q}(x)$ by noticing that (2.7) is a polynomial eigenvalue problem (PEP) [27] with matrix pencil

$$S^{p,q}(x^*) = \sum_{i=0}^{d} S^{(i)} B_i(x^*), \tag{2.8}$$

in the basis $\{B_i(x^*)\}$ and where $S^{(i)} \in \mathbb{C}^{n \times n}$ for $i = 0, \ldots, d$ and $d := \max(d_p^{(x)}, d_q^{(x)})$. It is well known that $p(x,y)$ and $q(x,y)$ share a nontrivial common factor if and only if $\mathrm{res}^{p,q}(x) \equiv 0$. In this case, the pencil (2.8) is said to be singular. If $p(x,y)$ and $q(x,y)$ don't share a common factor, the pencil is called regular. Unfortunately, the eigenvalues of a singular (or close to singular) pencil can be changed arbitrarily by small perturbations of its matrices [53]. However, the situation is not always as bad as this. Wilkinson gives the example of a singular pencil $A - \lambda B$, where

$$A := \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{and} \qquad B := \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The perturbed pencil $\hat{A} - \lambda\hat{B}$ has characteristic equation $\det(\hat{A} - \lambda\hat{B}) = [(2 + \Delta a_{11}) - \lambda(1 + \Delta b_{11})](\Delta a_{22} - \lambda \Delta b_{22}) - (\Delta a_{12} - \lambda \Delta b_{12})(\Delta a_{21} - \lambda \Delta b_{21})$, where $\hat{A} := A + \Delta A$ and $\hat{B} := B + \Delta B$. For almost all perturbations ΔA and ΔB, the characteristic equation has a root which is very close to 2 [53]. Hence, it may in some cases be possible to recover the eigenvalues of the regular part of a singular pencil. If this is not the case, the greatest common divisor of $p(x,y)$ and $q(x,y)$ or the singular part of the associated pencil should be extracted. Special care should be taken with both

approaches, as they will inevitably be based on decisions concerning the ranks of the matrices involved.

A standard way of solving the PEP (2.7) is to linearize the polynomial pencil $S^{p,q}(x^*)$ into a linear pencil $A - x^*B$ and solve its associated generalized eigenvalue problem (GEP). The matrices $A, B \in \mathbb{C}^{dn \times dn}$ are chosen so that $A - x^*B$ has the same spectrum as $S^{p,q}(x^*)$. In practice, the monomial basis is used most often together with a companion linearization. For the Chebyshev basis, a colleague linearization can be used. In fact, both of these are confederate linearizations [29,30], which are defined by the polynomial basis' recurrence relations. To construct such a linearization, consider as per example the recurrence relations

$$x \begin{bmatrix} 1 & x & \cdots \end{bmatrix} = \begin{bmatrix} 1 & x & \cdots \end{bmatrix} \cdot \begin{bmatrix} 0 & & \\ 1 & & \\ & \ddots & \end{bmatrix} \text{ and} \tag{2.9a}$$

$$x \begin{bmatrix} T_0(x) & T_1(x) & \cdots \end{bmatrix} = \begin{bmatrix} T_0(x) & T_1(x) & \cdots \end{bmatrix} \cdot \begin{bmatrix} 0 & \frac{1}{2} & & \\ 1 & 0 & \frac{1}{2} & \\ & \frac{1}{2} & 0 & \ddots \\ & & \frac{1}{2} & \ddots \\ & & & \ddots \end{bmatrix} \tag{2.9b}$$

for the monomial and Chebyshev basis, respectively. Let $\boldsymbol{v}^{(d)}(x)$ be the generalized Vandermonde vector $\begin{bmatrix} B_0(x) & \cdots & B_{d-1}(x) \end{bmatrix}^{\mathrm{T}}$ of length d and let $\boldsymbol{l}$ be a left eigenvector of the PEP (2.7) so that

$$\boldsymbol{l}^{\mathrm{T}} \cdot S^{p,q}(x^*) = \boldsymbol{0}. \tag{2.10}$$

Now truncate equations (2.9) up to the dth term and convert to block form by treating scalars as identity matrices, then use (2.10) to eliminate the last block row. Defining $\boldsymbol{w} := \boldsymbol{v}^{(d)}(x^*) \otimes \boldsymbol{l}$, we obtain the companion and colleague linearizations

$$x^* \boldsymbol{w}^{\mathrm{T}} \cdot \begin{bmatrix} \mathbb{I} & & & \\ & \ddots & & \\ & & \mathbb{I} & \\ & & & S^{(d)} \end{bmatrix} = \boldsymbol{w}^{\mathrm{T}} \cdot \begin{bmatrix} 0 & & & -S^{(0)} \\ \mathbb{I} & & & -S^{(1)} \\ & \ddots & & \vdots \\ & & \mathbb{I} & -S^{(d-1)} \end{bmatrix} \text{ and} \tag{2.11a}$$

$$2x^* \boldsymbol{w}^{\mathrm{T}} \cdot \begin{bmatrix} \mathbb{I} & & & \\ & \ddots & & \\ & & \mathbb{I} & \\ & & & S^{(d)} \end{bmatrix} = \boldsymbol{w}^{\mathrm{T}} \cdot \begin{bmatrix} 0 & \mathbb{I} & & & -S^{(0)} \\ 2\mathbb{I} & 0 & \ddots & & \vdots \\ & \mathbb{I} & \ddots & \mathbb{I} & -S^{(d-3)} \\ & & \ddots & 0 & S^{(d)} - S^{(d-2)} \\ & & & \mathbb{I} & -S^{(d-1)} \end{bmatrix} \tag{2.11b}$$

for the monomial and Chebyshev basis, respectively. Here, $\boldsymbol{w}$ is a left eigenvector of the GEP (2.11). If $(x^*, y^*) \in \mathbb{C}^2$ is a root of $p(x, y) = q(x, y) = 0$, then $\boldsymbol{v}^{(n)}(y^*)$ is a left eigenvector $\boldsymbol{l}$ of (2.10) corresponding to the eigenvalue x^* since

$$\boldsymbol{v}^{(n)}(y^*)^{\mathrm{T}} \cdot S^{p,q}(x^*) = \begin{bmatrix} p(x^*, y^*) \boldsymbol{v}^{(d_q^{(y)})}(y^*)^{\mathrm{T}} & q(x^*, y^*) \boldsymbol{v}^{(d_p^{(y)})}(y^*)^{\mathrm{T}} \end{bmatrix} = \boldsymbol{0}^{\mathrm{T}}.$$

This means that if x^* is a simple eigenvalue of the GEP (2.11), we can recover y^* from its corresponding left eigenvector $\boldsymbol{w}$, which will be a multiple of $\boldsymbol{v}^{(d)}(x^*) \otimes \boldsymbol{v}^{(n)}(y^*)$. However, the situation becomes more difficult when (x^*, y^*) is a multiple root, or when there is a $\hat{y}^* \neq y^*$ for which $(x^*, \hat{y}^*)$ is also a root. Both of these cases increase the dimension of the kernel of $S^{p,q}(x^*)$ so that it is larger than one. Several solutions for recovering all values y^* under these circumstances have been proposed.

Some algorithms compute y^* by solving a smaller eigenproblem for each distinct x^* [8, 11, 28]. However, this approach requires estimating the multiplicities of the eigenvalues x^*, along with the dimension of their corresponding kernels. The former task is complicated by the fact that a multiple real eigenvalue x^* often splits into several nearby complex eigenvalues due to round-off errors or because the problem cannot be represented exactly in finite precision. Then, for each distinct x^* an eigenproblem of size equal to the dimension of its kernel should be solved. Another approach is to eliminate both x and y separately and then for each pair of projected coordinates check whether it constitutes a solution of the given system or not [6,12,41]. For many applications, a nontrivial final task is to determine which of the solutions are real.

To solve these issues, we propose to first project the solutions onto the real plane associated with x and y, which will allow us to compute both the x- and y-coordinates of the roots simultaneously and ensure they are real. Moreover, this approach does not require computing any eigenvectors or solving any additional eigenproblems, which saves both computational effort and memory. Consider the change of variables

$$(x, y) = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix} \cdot (z, \omega) \tag{2.12}$$

which transforms the polynomials $p(x, y)$ and $q(x, y)$ into $\hat{p}(z, \omega)$ and $\hat{q}(z, \omega)$, respectively. This operation can destroy the coordinate degree structure of the polynomials, possibly changing the coefficient matrices from dense rectangular matrices to square upper anti-triangular matrices. However, the total degrees of the polynomials remain unchanged. We proceed as before, this time eliminating ω from the modified system $\hat{p}(z, \omega) = \hat{q}(z, \omega) = 0$. The roots of the resultant $\operatorname{res}^{\hat{p},\hat{q}}(z)$ now comprise the set

$$V_{\mathbb{C}}^{(z)} := \{z \in \mathbb{C} \,|\, \exists\, \omega \in \mathbb{C} : \hat{p}(z, \omega) = \hat{q}(z, \omega) = 0\}.$$

In fact, $\{z \in \mathbb{C} \,|\, \operatorname{res}^{\hat{p},\hat{q}}(z) = 0\}$ is often exactly equal to $V_{\mathbb{C}}^{(z)}$, since the transformation (2.12) generically gives rise to a constant leading coefficient system $\hat{p}_{d_p}(z) = \hat{q}_{d_q}(z) = 0$ with no solutions. Since the transformation in (2.12) is invertible, $V_{\mathbb{C}}^{(z)}$ can be characterized in terms of the roots of the original system $p(x, y) = q(x, y) = 0$ as

$$V_{\mathbb{C}}^{(z)} = \{z \in \mathbb{C} \,|\, \exists\, (x, y) \in V_{\mathbb{C}} : z = x + iy\}.$$

Clearly, the set of projected real roots

$$V_{\mathbb{R}^2}^{(z)} := \{z \in \mathbb{C} \,|\, \exists (x, y) \in V_{\mathbb{C}} \cap \mathbb{R}^2 : z = x + iy\}$$

is a subset of $V_{\mathbb{C}}^{(z)}$. As we are only interested in real roots $(x^*, y^*) \in \mathbb{R}^2$, we may also interpret the transformed system as a polyanalytic polynomial system $\hat{p}(z, \overline{z}) = \hat{q}(z, \overline{z}) = 0$. Hence, the real roots $(x^*, y^*) \in \mathbb{R}^2$ of a bivariate polynomial system may be obtained as the real and imaginary part of (a subset of) the eigenvalues of the GEP (2.11) associated with the elimination of $\overline{z}$ after converting the system

into a polyanalytic system with (2.12). In the same way, the complex roots $z^* \in \mathbb{C}$ of a polyanalytic polynomial system can be obtained by eliminating $\overline{z}$. For both types of systems, it remains to identify which of the eigenvalues do not correspond to roots of the original system. In the final subsection, we present a robust method to remove these spurious solutions based on a Newton–Raphson refinement on the original polynomial system. The following subsection proposes several balancing steps that aim to maximize the accuracy of the computed roots.

2.2. Balancing the system and its associated pencil.

2.2.1. Balancing the bivariate system. From here on, we restrict the discussion to polynomial systems given in the product monomial basis for simplicity. For both bivariate and polyanalytic polynomial systems, we center the exponents of the elements of the associated coefficient matrices P and Q around zero with the scaling

$$P \leftarrow 2^{-\mathrm{round}(\mathrm{median}(\log_2(|P|)))} P \tag{2.13a}$$

$$Q \leftarrow 2^{-\mathrm{round}(\mathrm{median}(\log_2(|Q|)))} Q. \tag{2.13b}$$

Here, $|\cdot|$ is the absolute value, $\log(\cdot)$ is the element-wise logarithm, $\mathrm{median}(\cdot)$ is the median over all finite elements of its argument and $\mathrm{round}(\cdot)$ rounds to the nearest integer to avoid introducing round-off error. The scaling (2.13) not only normalizes the coefficients of the two polynomials relative to each other, but also maximizes the available exponent range so that overflow is less likely to occur.

Bivariate systems $p(x, y) = q(x, y) = 0$ are then converted into polyanalytic polynomial systems $\hat{p}(z, \overline{z}) = \hat{q}(z, \overline{z}) = 0$ using (2.12). The entries of the coefficient matrices $\hat{P}$ and $\hat{Q}$ are linear combinations of the entries of the coefficient matrices P and Q. In order to minimize loss of precision due to rounding errors, we propose to first minimize the variance of the exponents of their elements by scaling the variables as $x \leftarrow 2^{r_x} x$ and $y \leftarrow 2^{r_y} y$ with

$$\hat{P} \leftarrow \mathrm{diag}(\boldsymbol{v}^{(d_p^{(y)}+1)}(2^{r_y})) \cdot P \cdot \mathrm{diag}(\boldsymbol{v}^{(d_p^{(x)}+1)}(2^{r_x})) \tag{2.14a}$$

$$\hat{Q} \leftarrow \mathrm{diag}(\boldsymbol{v}^{(d_q^{(y)}+1)}(2^{r_y})) \cdot Q \cdot \mathrm{diag}(\boldsymbol{v}^{(d_q^{(x)}+1)}(2^{r_x})) \tag{2.14b}$$

where r_x and r_y are real scalars which solve the least squares problem

$$\underset{r_x, r_y \in \mathbb{R}}{\text{minimize}} \quad \mathrm{var}(\log(|\hat{P}|)) + \mathrm{var}(\log(|\hat{Q}|)), \tag{2.15}$$

in which $\mathrm{var}(\cdot)$ is the sample variance over all finite elements of its argument. Again, we round r_x and r_y to their nearest integer values to avoid introducing round-off error. After applying (2.14), we transform $\hat{P}$ and $\hat{Q}$ in-place with (2.12).

2.2.2. Balancing the polyanalytic system. Assuming we now dispose of a scaled polyanalytic system $\hat{p}(z, \overline{z}) = \hat{q}(z, \overline{z}) = 0$, our goal is to balance the problem such that its associated pencil $A - z^* B$ is closer to a well-conditioned normal pencil [25]. Ward suggests to scale all elements of A and B to the same order of magnitude [51]. To this end, a simple heuristic is to minimize the variance of the exponents of their elements by combining two balancing strategies. The first is to balance the coefficient matrices by scaling the variables z and $\overline{z}$ with a real scalar s_z as $z \leftarrow 2^{s_z} z$ and $\overline{z} \leftarrow 2^{s_z} \overline{z}$. The second is to balance the Sylvester blocks $S^{(i)}$ by scaling the

variable z^* in the PEP (2.7) with a real scalar s_{z^*} as $z^* \leftarrow 2^{s_{z^*}} z^*$. We set

$$\hat{P} \leftarrow \operatorname{diag}(\boldsymbol{v}^{(d_{\hat{P}}^{(\overline{z})}+1)}(2^{s_z})) \cdot \hat{P} \cdot \operatorname{diag}(\boldsymbol{v}^{(d_{\hat{P}}^{(z)}+1)}(2^{s_z+s_{z^*}})) \tag{2.16a}$$

$$\hat{Q} \leftarrow \operatorname{diag}(\boldsymbol{v}^{(d_{\hat{Q}}^{(\overline{z})}+1)}(2^{s_z})) \cdot \hat{Q} \cdot \operatorname{diag}(\boldsymbol{v}^{(d_{\hat{Q}}^{(z)}+1)}(2^{s_z+s_{z^*}})) \tag{2.16b}$$

Analogously to (2.15), s_z and s_{z^*} are the solution of the least squares problem

$$\underset{s_z, s_{z^*} \in \mathbb{R}}{\text{minimize}} \quad \operatorname{var}(\log(|\hat{P}|)) + \operatorname{var}(\log(|\hat{Q}|)). \tag{2.17}$$

Again, we round the solution of (2.17) to the nearest integer before applying transformations (2.16) to avoid introducing round-off error. Finally, we center the exponents of the coefficient matrices by applying (2.13) to $\hat{P}$ and $\hat{Q}$.

2.2.3. Balancing the pencil. Solving the GEP (2.11) consists of three steps. Let A and B be the matrices in the right and left hand side of (2.11), respectively. The first step is to reduce B to an upper triangular matrix with a QR-factorization. Then, the pair (A, B) is reduced to generalized upper Hessenberg form using unitary left and right transformations so that A becomes an upper Hessenberg matrix and B remains upper triangular. Finally, the QZ algorithm is applied on the resulting pencil. Equation (2.11) is not the only way to linearize the PEP (2.7). However, compared to other linearizations it has the advantage that the pencil is already in a block generalized Hessenberg form. Moreover, many of the first Givens rotations reducing the pencil to a generalized Hessenberg form will be permutations. An alternative linearization is the second companion form [16], which is perhaps less efficient but in our experience has a slight edge in accuracy over (2.11).

To improve the conditioning of the eigenvalues of the pencil $A - z^*B$, most balancing strategies are based on a two-sided diagonal transformation. The LAPACK [1] routine ZGGBAL implements Ward's algorithm [51], which aims to scale the elements of A and B such that their magnitudes are as close to unity as possible. Unfortunately, LAPACK's driver routine ZGGEV for computing the generalized eigenvalues of a pair of complex nonsymmetric matrices (and, consequently, MATLAB) disables Ward's algorithm by default. More recently, Lemonnier and Van Dooren [25] suggest to (approximately) solve the convex optimization problem

$$\underset{\det(D_l \cdot D_r)=1}{\text{minimize}} \quad \|D_l \cdot A \cdot D_r\|_{\mathrm{F}}^2 + \|D_l \cdot B \cdot D_r\|_{\mathrm{F}}^2 \tag{2.18}$$

for the real positive diagonal matrices D_l and D_r. In fact, this objective corresponds to searching for D_l and D_r so that $D_l^2 \cdot (|A|^2 + |B|^2) \cdot D_r^2$ is a doubly stochastic matrix, i.e., having row and column sums equal to one. The Sinkhorn–Knopp (SK) algorithm is perhaps the most simple method for finding a doubly stochastic scaling of a nonnegative matrix. Its simplicity has led to the repeated discovery of the method [21], of which the algorithm proposed in [25] is another variation. Given a nonnegative matrix $M := |A|^2 + |B|^2$, SK alternately solves for D_l and D_r by fixing D_r (D_l) and then setting the column (row) sums equal to one with the iterates

$$D_l^2 \leftarrow \operatorname{diag}(M \cdot D_r^2 \cdot \mathbf{1}_{dn \times 1})^{-1} \text{ and} \tag{2.19a}$$

$$D_r^2 \leftarrow \operatorname{diag}(\mathbf{1}_{1 \times dn} \cdot D_l^2 \cdot M)^{-1}, \tag{2.19b}$$

respectively. The main cost of the SK iterates (2.19) are the left and right matrix-vector products with M, which are $O(d^2n^2)$ floating point operations (flop) each.

Depending on the number of iterations, SK can be close to as expensive as solving the GEP (2.11) itself, which costs $O(d^3n^3)$ flop. To reduce the impact of SK on the total computational time, the block structure in M can be exploited in the matrix-vector products. Taking only the block structure and identity matrices into account reduces the computational complexity to $O(dn^2)$ flop per matrix-vector product, which is small enough in practice. Before applying D_l and D_r, their entries are rounded to the nearest power of two so that the transformation does not introduce round-off error.

2.3. Filtering spurious solutions with Newton–Raphson. Removing spurious solutions introduced by the resultant or by round-off error is not as easy as testing if the polynomials evaluated at the candidate roots are (close to) zero, even after normalization. This is because evaluating a polynomial may be, and often is, ill-conditioned, especially as the modulus of the candidate root increases. Even taking the condition number into account does not improve such a filter's robustness much. We propose to refine the roots with a (complex) Newton–Raphson algorithm [44], and simultaneously remove spurious solutions based on the step length. For candidate roots close to a root of the polynomial system, the step length will be small relative to the modulus of the root, while the step length for spurious roots tends to be large relative to its modulus.

Writing the bivariate polynomial system $p(x, y) = q(x, y) = 0$ as a vector-valued residual function $F^{p,q}(x, y) := (p(x, y), q(x, y))$, the Newton–Rapshon step $(\Delta x, \Delta y)$ for a candidate root (x^*, y^*) is computed as

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = (J^{p,q}(x^*, y^*))^\dagger \cdot F^{p,q}(x^*, y^*), \tag{2.20a}$$

where $J^{p,q}$ is the Jacobian $\partial F^{p,q}/\partial(x, y)^{\mathrm{T}}$. The next iterate is then $(x^*, y^*) \leftarrow (x^*, y^*) + (\Delta x, \Delta y)$.

In the polyanalytic case $p(z, \overline{z}) = q(z, \overline{z}) = 0$, the standard way of computing the Newton–Raphson step at a candidate root z^* is to write $p(z, \overline{z}) = u_p(z, \overline{z}) + iv_p(z, \overline{z})$ and $q(z, \overline{z}) = u_q(z, \overline{z}) + iv_q(z, \overline{z})$, wherein u_p, v_p, u_q and v_q are real-valued polyanalytic polynomials. Define the vector-valued residual function $F_{\mathbb{R}}^{u_p,u_q,v_p,v_q}(z, \overline{z}) := (u_p(z, \overline{z}), u_q(z, \overline{z}), v_p(z, \overline{z}), v_q(z, \overline{z}))$ and let x and y be the real and imaginary part of z, respectively. The Newton–Raphson step $(\Delta x, \Delta y)$ at a point z^* can then be computed as

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \left(J_{\mathbb{R}}^{u_p,v_p,u_q,v_q}(z^*, \overline{z}^*)\right)^\dagger \cdot F_{\mathbb{R}}^{u_p,u_q,v_p,v_q}(z^*, \overline{z}^*), \tag{2.20b}$$

where $J_{\mathbb{R}}^{u_p,u_q,v_p,v_q}$ is the real Jacobian $\partial F_{\mathbb{R}}^{u_p,u_q,v_p,v_q}/\partial(x, y)^{\mathrm{T}}$. With a step of this form, the next iterate is $z^* \leftarrow z^* + (\Delta x + i\Delta y)$. However, obtaining u_p, v_p, u_q and v_q and their partial derivatives given only $p(z, \overline{z})$ and $q(z, \overline{z})$ is cumbersome. The real Jacobian may be obtained much more elegantly from a complex Taylor series of the complex vector-valued residual function $F^{p,q}(z, \overline{z}) := (p(z, \overline{z}), q(z, \overline{z}))$. First, we compute its complex Jacobian

$$J_{\mathbb{C}}^{p,q} := \begin{bmatrix}\dfrac{\partial p}{\partial z} & \dfrac{\partial p}{\partial \overline{z}}\\ \dfrac{\partial q}{\partial z} & \dfrac{\partial q}{\partial \overline{z}}\end{bmatrix},$$

where the partial derivatives are so-called Wirtinger derivatives and are with respect to z ($\overline{z}$), while treating $\overline{z}$ (z) as constant. By defining

$$J_{\mathbb{R}}^{p,q} := J_{\mathbb{C}}^{p,q} \cdot \begin{bmatrix} 1 & i \\ 1 & -i \end{bmatrix},$$

the real Jacobian can then be recovered from the identity [44]

$$J_{\mathbb{R}}^{u_p,v_p,u_q,v_q} \equiv \begin{bmatrix} \mathrm{Re}\{J_{\mathbb{R}}^{p,q}\} \\ \mathrm{Im}\{J_{\mathbb{R}}^{p,q}\} \end{bmatrix}. \tag{2.21}$$

The advantage of (2.21) is that evaluating $J_{\mathbb{C}}^{p,q}$ requires only four complex function evaluations, compared to eight real function evaluations for $J_{\mathbb{R}}^{u_p,u_q,v_p,v_q}$ together with the cost of constructing of the four polynomials u_p, v_p, u_q and v_q. It is interesting to note that $(J_{\mathbb{R}}^{p,q}(z^*, \overline{z}^*))^\dagger \cdot F^{p,q}(z^*, \overline{z}^*)$ computes a complex Newton–Raphson step, while the form (2.20b) ensures that the step will be real.

It is recommended to solve the least squares systems (2.20) either by regularizing the system or by approximating the Moore–Penrose pseudo-inverse using the singular value decomposition (SVD). This is especially important near multiple roots, where the Jacobian tends to a singular matrix. Since the Jacobian is a small matrix of size 2×2 or 4×2, we choose the SVD and invert the first component if $\sigma_1 > \tau_\sigma$ and additionally invert the second component if $\sigma_2 > \sigma_1 \tau_\sigma$, where σ_1 and σ_2 are the first two singular values of the Jacobian and τ_σ is a user-defined tolerance.

Next, we describe a method to remove spurious roots based on the Newton–Raphson step length which we have found to work well in practice. Let $(\Delta x, \Delta y)$ be a solution of (2.20a) or (2.20b) at a candidate root (x^*, y^*) or z^*, respectively. If the candidate root is close to a root of the polynomial system, it may be expected that its Newton–Raphson step will be small in comparison to a measure of the size of the root. Based on this observation, we identify a candidate root of a bivariate system as spurious if

$$\|(\Delta x, \Delta y)\| \geq \tau_{\mathrm{NR}} \cdot \max(\|(x^*, y^*)\|, 1) \tag{2.22a}$$

and of a polyanalytic system if

$$\|(\Delta x, \Delta y)\| \geq \tau_{\mathrm{NR}} \cdot \max(|z^*|, 1) \tag{2.22b}$$

holds for some tolerance τ_{NR}. This condition corresponds to a relative tolerance on the step length outside the unit circle, and an absolute tolerance inside the unit circle. The smaller τ_{NR}, the less likely spurious roots are returned as solutions, but also the more likely valid roots are discarded.

In summary, we describe the complete process for solving bivariate and polyanalytic univariate polynomial systems in Algorithm 2.1. In the following section, we investigate the performance of this and other algorithms on a set of bivariate systems, a subset of which are considered difficult due to the wide variation in exponents of the polynomial's coefficients. We will see that Algorithm 2.1 is not only able to recover the largest fraction of roots, but also does this accurately and efficiently.

3. Numerical experiments. We compare the relative performance of Algorithm 2.1, Chebfun2 alpha 2923 [47], PHCpack 2.3.79 [49], the RealSolving C library [38,39], the RegularChains package [2] and Maple's BivariatePolynomial method

Input: Two bivariate or polyanalytic univariate polynomials p and q
Ouput: Isolated real roots (x^*, y^*) or complex roots z^* of the system $p = q = 0$

$\{P, Q\} \leftarrow$ Apply (2.13) to center the exponents of the coefficients
if *p and q are bivariate* **then**
  $\{r_x, r_y\} \leftarrow$ Solve (2.15)
  $\{r_x, r_y\} \leftarrow \{\text{round}(r_x), \text{round}(r_y)\}$
  $\{\hat{P}, \hat{Q}\} \leftarrow$ Apply the transformation (2.14) to balance the system
  $\{\hat{P}, \hat{Q}\} \leftarrow$ Apply the transformation (2.12) to obtain a polyanalytic system
end
$\{s_z, s_{z^*}\} \leftarrow$ Solve (2.17)
$\{s_z, s_{z^*}\} \leftarrow \{\text{round}(s_z), \text{round}(s_{z^*})\}$
$\{\hat{P}, \hat{Q}\} \leftarrow$ Apply the transformation (2.16) to balance the polyanalytic system
$\{\hat{P}, \hat{Q}\} \leftarrow$ Apply (2.13) to $\{\hat{P}, \hat{Q}\}$ to center the exponents of the coefficients
$A - z^*B \leftarrow$ Build (2.11) associated with eliminating $\overline{z}$ from $\hat{p} = \hat{q} = 0$
$\{D_l, D_r\} \leftarrow$ Approximately solve (2.18) with SK
$\{D_l, D_r\} \leftarrow \{\text{diag}(2^{\text{round}(\log_2(\text{diag}(D_l)))}), \text{diag}(2^{\text{round}(\log_2(\text{diag}(D_r)))})\}$
$\boldsymbol{z}^* \leftarrow \text{eig}(D_l \cdot A \cdot D_r, D_l \cdot B \cdot D_r)$
foreach *z^* in $\boldsymbol{z}^*$* **do**
  if *p and q are bivariate* **then**
    $(x^*, y^*) \leftarrow$ Decode $z^* = x^* + iy^*$
  end
  for $i = 1$ **to** maxiter *(e.g.,* maxiter $= 4$*)* **do**
    $(\Delta x, \Delta y) \leftarrow$ Solve (2.20) at (x^*, y^*) or z^* (for, e.g., $\tau_\sigma = 10^{-6}$)
    if *(2.22) holds (for, e.g., $\tau_{\text{NR}} = 10^{-2}$)* **then**
      Discard (x^*, y^*) or z^* and break
    end
    if *p and q are bivariate* **then**
      $(x^*, y^*) \leftarrow (x^*, y^*) + (\Delta x, \Delta y)$
    else
      $z^* \leftarrow z^* + (\Delta x + i\Delta y)$
    end
  end
end

Algorithm 2.1: Compute isolated roots of a bivariate or polyanalytic univariate polynomial system.

on a wide range of bivariate polynomial systems using the performance profiles proposed by Dolan and Moré [13]. Algorithm 2.1 was implemented as part of the MATLAB toolbox Tensorlab [45]. Chebfun2 currently has two algorithms for computing the common roots of two bivariate functions. In our experiments, we use the recently added method based on the Bézout resultant [34]. Chebfun2 is the only software package in this comparison which chooses to represent the functions, in this case polynomials, in the product Chebyshev basis. PHCpack is a well-known homotopy continuation solver. The RealSolving library is based on reducing the polynomial system to a rational univariate representation, from which the solutions can easily be recovered. The RegularChains package computes a triangular decomposition of

the system, i.e., a set of simpler systems called regular chains which represent the solutions of the original system. Similar to but less advanced than Chebfun2, Maple's BivariatePolynomial algorithm applies a random orthogonal change of variables, computes the y-coordinates of the isolated roots as the eigenvalues of the polynomial matrix associated with the Bézout resultant and recovers the x-coordinates from the corresponding eigenvectors. The first two algorithms were applied in MATLAB 8.1 (R2013a), PHCpack 2.3.79 as a standalone executable, and the remaining three methods using Maple 17. All experiments were performed on a server with two hexacore Intel Xeon E5645 CPUs and 48 GB RAM.

TABLE 3.1
Properties of the initial low-degree problem set, containing a total of 49 systems.

Problem	**Total degree**	**Exponent range**
`ex001`–`ex025` [8]	$(3,2)$ to $(11,10)$	$(0.30,0.00)$ to $(8.79,1.71)$
`C1`–`C4`, `D1`–`D2`, `M1`–`M4`, `R1`–`R3` [12]	$(2,2)$ to $(16,5)$	$(0.00,0.00)$ to $(8.05,8.65)$
`Q21`–`Q210`, `X` [37]	$(3,3)$	$(0.60,0.42)$ to $(2.40,2.47)$

TABLE 3.2
Properties of the initial moderate-degree problem set, containing a total of 16 systems. All of these systems with the exception of `lebesgue` *were taken from [6].*

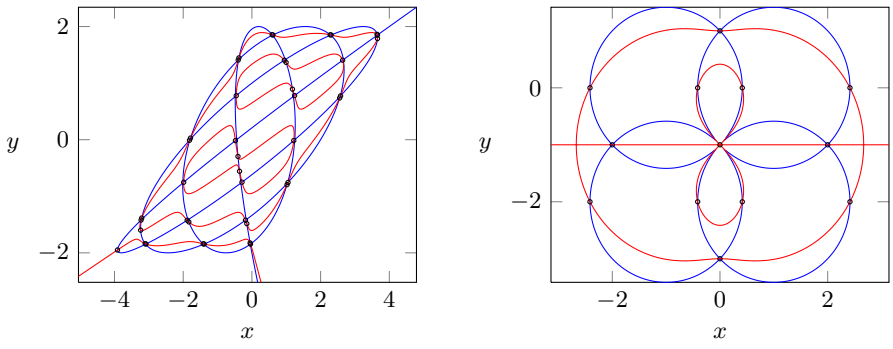
Problem	**Total degree**	**Exponent range**
`13_sings_9`	$(9,8)$	$(6.91,6.26)$
`compact_surf`	$(18,17)$	$(3.76,3.76)$
`curve24`	$(29,28)$	$(4.95,4.68)$
`curve_issac`	$(16,15)$	$(3.43,4.24)$
`cusps_and_flexes`	$(9,8)$	$(4.74,4.29)$
`deg16_7_curves`	$(32,11)$	$(3.16,1.08)$
`dfold_10_6`	$(32,31)$	$(4.97,4.46)$
`grid_deg_10`	$(10,9)$	$(9.08,8.16)$
`hard_one`	$(27,10)$	$(28.04,7.82)$
`huge_cusp`	$(8,7)$	$(4.79,4.16)$
`L4_circles`	$(16,15)$	$(8.05,8.65)$
`lebesgue`	$(20,20)$	$(6.22,6.53)$
`spiral29_24`	$(29,29)$	$(10.94,4.95)$
`ten_circles`	$(20,19)$	$(6.10,5.57)$
`tryme`	$(34,25)$	$(7.68,6.96)$
`vert_lines`	$(16,14)$	$(2.15,2.14)$

Each algorithm's performance is evaluated on two problem sets. The first set is generated using the low-degree bivariate polynomial systems listed in Table 3.1. The second set is more difficult and is generated using the bivariate polynomial systems listed in Table 3.2. The zero level lines and their intersections of two example systems of each type are displayed in Figure 3.1 and Figure 3.2. Both tables show the total degrees of the system and the exponent ranges of the coefficients. For a polynomial system $p(x,y) = q(x,y) = 0$ the total degree is displayed as (d_p, d_q), while the

exponent range is defined as $(\text{range}(p(x,y)), \text{range}(q(x,y)))$, where

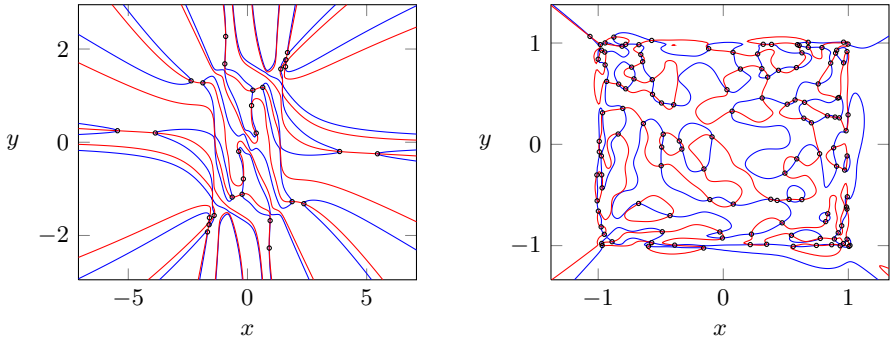
$$\text{range}(p(x,y)) := \max_{i,j} \log_{10}(|p_{ij}|) - \min_{\substack{i,j \\ p_{ij}\neq 0}} \log_{10}(|p_{ij}|).$$

The total degrees define the size of the eigenproblem, which is generically equal to $\max(d_p, d_q)(d_p + d_q)$. Large exponent ranges are usually associated with difficult problems since round-off errors tend to increase in magnitude as the exponent range increases.



(a) `ex007` (b) `C3`

FIG. 3.1. *Two examples of the low-degree bivariate polynomial systems and their solutions.*



(a) `curve24` (b) `lebesgue`

FIG. 3.2. *Two examples of the moderate-degree bivariate polynomial systems and their solutions.*

All systems are given in the product monomial basis, often with integer coefficients. In some cases these coefficients contain more digits than can be represented by double precision floating point numbers. For a fair comparison, each algorithm receives the systems in a normalized format. More specifically, each polynomial's coefficients are converted to hardware double precision floating point numbers, after which they are multiplied by an integer power of two such that their exponents are approximately centered around zero.

Some of the systems are of the form $p(x,y) = \frac{\partial p}{\partial y}(x,y) = 0$ and correspond to finding roots of $p(x,y)$ that are tangent to a vertical line. Taking inspiration from

this, we artificially expand the low-degree and medium-degree problem sets by taking each system $p(x,y) = q(x,y) = 0$ and adding the systems $p(x,y) = \frac{\partial p}{\partial x}(x,y) = 0$ and $\frac{\partial p}{\partial x}(x,y) = \frac{\partial p}{\partial y}(x,y) = 0$, tripling the total number of systems. The only exception to this rule are the systems `Q21`–`Q210` and `X`, which are already of the form $\frac{\partial p}{\partial x}(x,y) = \frac{\partial p}{\partial y}(x,y) = 0$ and correspond to finding the stationary points of normal quartics [37].

For each system in the expanded low-degree and moderate-degree problem sets, we compute a reference solution set as follows. First, all solutions of all systems as computed by all six algorithms are collected. There is no discrimination between real or complex solutions and valid or spurious solutions. For each such candidate root (x^*, y^*), we apply a high-precision Newton–Raphson method with 128 significant decimal digits to $\mathrm{Re}\{(x^*, y^*)\}$. We terminate the refinement after 500 iterations or when the relative step size is less than $\epsilon_{\mathrm{mach}} 10^{-3}$, where ϵ_{mach} is the machine epsilon for double precision. The refined root (x^*, y^*) is accepted as a valid root if both conditions

$$|p(x^*, y^*)| \leq |p|(|x^*|, |y^*|)\epsilon_{\mathrm{mach}} 10^3 \text{ and}$$
$$|q(x^*, y^*)| \leq |q|(|x^*|, |y^*|)\epsilon_{\mathrm{mach}} 10^3,$$

hold when evaluated in high-precision. Here, the polynomials $|p|$ and $|q|$ correspond to the coefficient matrices $|P|$ and $|Q|$, respectively. Evaluated at the absolute value of (x^*, y^*), they are equal to the condition number of evaluating the polynomials p and q at (x^*, y^*). If accepted, the candidate root is truncated to double precision and added to the system's reference solution set. After all candidate roots have been processed, the reference solution sets are reduced to their distinct elements.

For each problem set P, the performance profiles are computed as follows. Let S be the set of solvers consisting of the previously mentioned six algorithms and let $t_{p,s}$ denote the time required by solver s to solve problem p. If solver s did not successfully solve problem p, we set $t_{p,s} = \infty$. The performance profile for solver s,

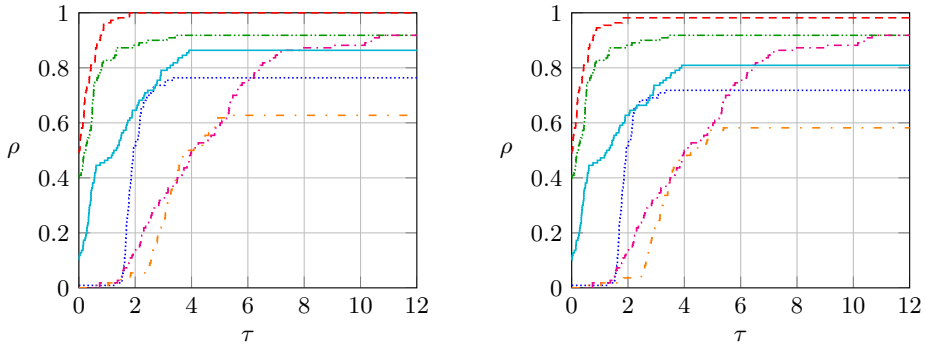
$$\rho_s(\tau) := \frac{|\{p \in P \,|\, t_{p,s} \leq 2^\tau \cdot (\min_{s \in S} t_{p,s})\}|}{|P|},$$

depicts the fraction of problems that solver s was able to solve within a factor 2^τ of the fastest solution time for that problem. We say that a polynomial system p has been solved successfully by solver s if for each root (x^*, y^*) in the reference solution set, there is a candidate root $(\hat{x}^*, \hat{y}^*)$ computed by s which satisfies

$$\|(x^*, y^*) - (\hat{x}^*, \hat{y}^*)\| \leq \max(\|(x^*, y^*)\|, 1)\epsilon_{\max}$$

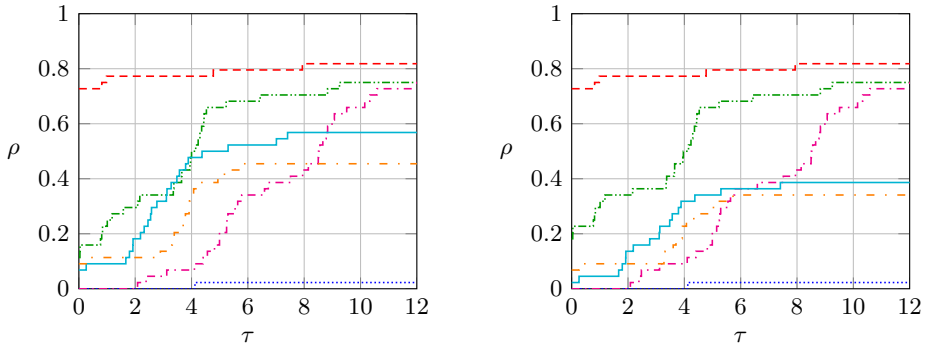
for a certain maximal relative error $\epsilon_{\max}$. Chebfun2 computes roots on a rectangular domain specified by the user, which is $[-1, 1] \times [-1, 1]$ by default. For each system, we set this domain equal to the smallest axis-aligned bounding box of the corresponding reference solution set, plus a small margin to prevent edge cases.

Figures 3.3 and 3.4 show the performance profiles of the expanded low-degree and moderate-degree problem sets, respectively, for two choices of $\epsilon_{\max}$. The left hand side, where $\tau = 0$, of each plot shows the fraction of problems for which a solver was the fastest solver. The right hand side, where τ is large, shows the fraction of problems a solver was able to successfully solve. For both problem sets and choices of $\epsilon_{\max}$, Algorithm 2.1 was able to successfully solve the largest fraction of problems and was also considerably faster than most other algorithms, with the RealSolving



(a) $\epsilon_{\max} = 5\%$ (b) $\epsilon_{\max} = 0.5\%$

FIG. 3.3. *Performance profiles for low-degree bivariate polynomial systems. The problem set consists of all these polynomial systems, and we say a system has been successfully solved if all of its roots have been found with a relative error that does not exceed* $\epsilon_{\max}$*. Legend: our method (----), PHCpack (——), Chebfun2 (- - -), RealSolving (-·-·-), RegularChains (-··-··-) and BivariatePolynomial (·····).*



(a) $\epsilon_{\max} = 5\%$ (b) $\epsilon_{\max} = 0.5\%$

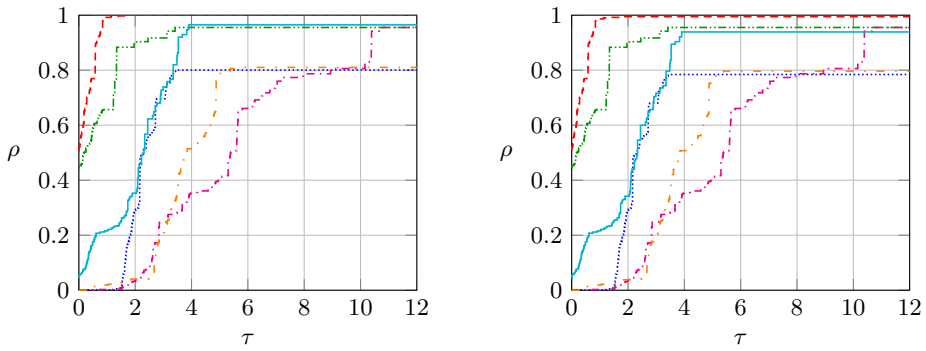
FIG. 3.4. *Performance profiles for moderate-degree bivariate polynomial systems. The problem set consists of all these polynomial systems, and we say a system has been successfully solved if all of its roots have been found with a relative error that does not exceed* $\epsilon_{\max}$*. Legend: our method (----), PHCpack (——), Chebfun2 (- - -), RealSolving (-·-·-), RegularChains (-··-··-) and BivariatePolynomial (·····).*

library as a close second for the low-degree problem set. Concerning the accuracy of the computed solutions, PHCpack and Chebfun2 seem to be sensitive to the tolerance $\epsilon_{\max}$, especially on the moderate-degree problem set. The other algorithms' performance profiles remain relatively static when $\epsilon_{\max}$ is decreased, indicating that their computed solutions are relatively more accurate.

Compared to the low-degree problem set, the fraction of moderate-degree systems that were successfully solved is much lower. These systems often have many more solutions than their low-degree counterparts. Since a solver by definition fails to solve a system as soon as just one of these roots is not found, we compute a second set of performance profiles to examine the fraction of successfully recovered roots as follows. We define two new problem sets containing all reference solutions of the expanded

low-degree systems and moderate-degree systems, respectively. The time for solver s to solve problem p is then defined as the total time for solver s to compute all roots of the bivariate polynomial system associated with p, divided by the number of roots of the system's reference solution set. If solver s was not able to successfully recover the root p, we set $t_{p,s} = \infty$. The condition for successfully recovering a root is identical to that of the previous performance profiles.

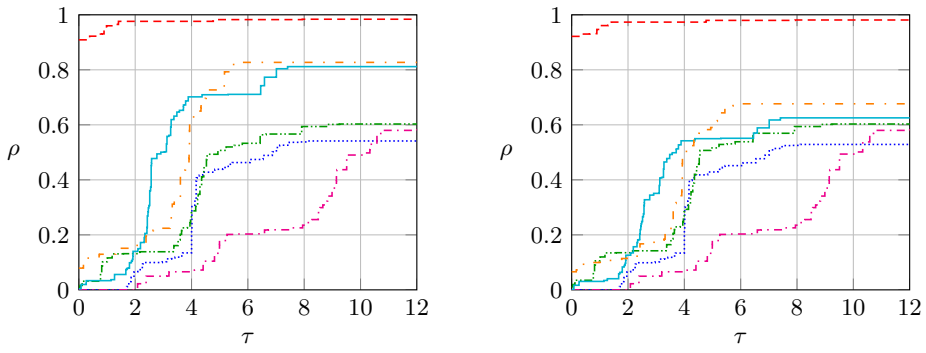
The performance profiles based on the redefined problem sets corresponding to the expanded low-degree and moderate-degree systems are shown in Figure 3.5 and 3.6, respectively. Algorithm 2.1 was able to recover nearly all roots of both problem sets and for both values of ϵ_{mach}. The roots that it did not recover, were spread out among different systems, causing the fraction of successfully solved moderate-degree systems to drop to around 80% (cf. Figure 3.4). Interestingly, the RealSolving library successfully solved the second largest fraction of systems, yet PHCpack and Chebfun2 successfully recovered the second largest fraction of roots in Figure 3.6(a). This indicates that the RealSolving library is somewhat of a hit or miss algorithm, either recovering all solutions or very few. In contrast, PHCpack and Chebfun2 can recover a larger fraction of roots, but fail to recover at least one root for many moderate-degree systems. Furthermore, we observe that for moderate-degree systems around 20% of the roots computed by PHCpack and Chebfun2 are only accurate up to $\epsilon_{\max} = 5\%$.



(a) $\epsilon_{\max} = 5\%$ (b) $\epsilon_{\max} = 0.5\%$

FIG. 3.5. *Performance profiles for low-degree bivariate polynomial systems. The problem set consists of all distinct roots of these systems, and we say a root has been successfully found if its relative error does not exceed* $\epsilon_{\max}$. *Legend: our method (----), PHCpack (——), Chebfun2 (- -), RealSolving (-···-), RegularChains (-·-·-) and BivariatePolynomial (·····).*

4. Conclusion. We presented an algorithm for computing the isolated real solutions of bivariate polynomial systems, and the isolated complex solutions of polyanalytic polynomial systems. For bivariate systems in two variables, say x and y, the algorithm projects the solutions onto the real plane associated with x and y by combining a well chosen change of variables together with a resultant-based elimination of a hidden variable. In this way, the problem is reduced to computing the eigenvalues of a polynomial eigenvalue problem. In comparison to other eigen-based approaches, there is no need to compute the accompanying eigenvectors or solve any additional eigenproblems. To increase the algorithm's robustness, we presented three methods of balancing the eigenvalue problem and an elegant method to remove spurious roots. Though the algorithm was chiefly developed for the polynomials expressed



(a) $\epsilon_{\max} = 5\%$ (b) $\epsilon_{\max} = 0.5\%$

FIG. 3.6. *Performance profiles for moderate-degree bivariate polynomial systems. The problem set consists of all distinct roots of these systems, and we say a root has been successfully found if its relative error does not exceed* $\epsilon_{\max}$. *Legend: our method (----), PHCpack (——), Chebfun2 (- · -), RealSolving (-··-), RegularChains (-·-·-) and BivariatePolynomial (·····).*

in the product monomial basis, the generalization to other bases is straightforward. The numerical experiments show that our method often outperforms other algorithms such as Chebfun2 and PHCpack in the fraction of roots successfully recovered, the accuracy of the estimated roots and computational time. The algorithms discussed in this article were implemented as part of the MATLAB toolbox Tensorlab [45] and are available online.

REFERENCES

[1] E. ANDERSON, Z. BAI, CHR. H. BISCHOF, S. BLACKFORD, J. W. DEMMEL, J. J. DONGARRA, J. J. DU CROZ, A. GREENBAUM, S. J. HAMMARLING, A. MCKENNEY, AND D. C. SORENSEN, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, third ed., 1999.

[2] P. AUBRY, D. LAZARD, AND M. MORENO MAZA, *On the theories of triangular sets*, J. Symb. Comput., 28 (1999), pp. 105–124.

[3] W. AUZINGER AND H. J. STETTER, *An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations*, in Proceedings of the International Conference on Numerical Mathematics, vol. 86 of Intern. Series in Numer. Math., Birkhäuser, Basel, 1988, pp. 12–30.

[4] C. BAJAJ, T. GARRITY, AND J. WARREN, *On the applications of multi-equational resultants*, Technical Report 88-826, Department of Computer Science, Purdue University, Nov. 1988.

[5] S. BASU, R. POLLACK, AND M.-F. ROY, *Algorithms in Real Algebraic Geometry*, vol. 10 of Algorithms and Computation in Mathematics, Springer-Verlag, second ed., 2006.

[6] E. BERBERICH, P. EMELIYANENKO, AND M. SAGRALOFF, *An elimination method for solving bivariate polynomial systems: Eliminating the usual drawbacks*, in Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX), M. Müller-Hannemann and R. Werneck, eds., San Francisco, CA, USA, Jan. 2011, SIAM, pp. 35–47.

[7] B. BUCHBERGER, *Gröbner bases and systems theory*, Multidim. Systems and Signal Proc., 12 (2001), pp. 223–251.

[8] 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, vol. 3718 of Lecture Notes in Computer Science, Springer-Verlag, 2005, pp. 75–92.

[9] 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 (1988), pp. 247–263.

[10] J. D. Carroll and J.-J. Chang, *Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart–Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.
[11] R. M. Corless, P. M. Gianni, and B. M. Trager, *A reordered Schur factorization method for zero-dimensional polynomial systems with multiple roots*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM Press, 1997, pp. 133–140.
[12] D. I. Diochnos, I. Z. Emiris, and E. P. Tsigaridas, *On the asymptotic and practical complexity of solving bivariate systems over the reals*, J. Symb. Comput., 44 (2009), pp. 818–835.
[13] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[14] I. Z. Emiris and B. Mourrain, *Matrices in elimination theory*, J. Symb. Comput., 28 (1999), pp. 3–43.
[15] J.-C. Faugère, *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)*, in Proceedings of the 2002 international symposium on Symbolic and algebraic computation, ISSAC '02, ACM, 2002, pp. 75–83.
[16] I. C. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.
[17] L. Gonzales-Vega and I. Necula, *Efficient topology determination of implicitly defined algebraic plane curves*, Comput. Aided Geom. Design, 19 (2002), pp. 719–743.
[18] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 84–84.
[19] G. F. Jónsson and S. A. Vavasis, *Accurate solution of polynomial equations using macaulay resultant matrices*, Math. Comp., 74 (2004), pp. 221–262.
[20] J. T. Kajiya, *Ray tracing parametric patches*, ACM SIGGRAPH Comp. Graph., 16 (1982), pp. 245–254.
[21] Ph. A. Knight, *The Sinkhorn-Knopp algorithm: Convergence and applications*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 261–275.
[22] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[23] J. B. Lasserre, *Global optimization with polynomials and the problems of moments*, SIAM J. Optim., 11 (2001), pp. 796–817.
[24] D. Lazard, *Résolution des systèmes d'équations algébriques*, Theoret. Comput. Sci., 15 (1981), pp. 77–110.
[25] D. Lemonnier and P. Van Dooren, *Balancing regular matrix pencils*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 253–263.
[26] T.-Y. Li, *Numerical solution of multivariate polynomial systems by homotopy continuation methods*, Act. Num., 6 (1997), pp. 399–436.
[27] D. Manocha, *Solving systems of polynomial equations*, IEEE Comput. Graph. Appl., 14 (1994), pp. 46–55.
[28] D. Manocha and J. W. Demmel, *Algorithms for intersecting parametric and algebraic curves II: Multiple intersections*, Graph. Models and Image Proc., 57 (1995), pp. 81–100.
[29] J. Maroulas and S. Barnett, *Polynomials with respect to a general basis. I. Theory*, J. Math. Anal. Appl., 72 (1979), pp. 177–194.
[30] ———, *Polynomials with respect to a general basis. II. Applications*, J. Math. Anal. Appl., 72 (1979), pp. 599–614.
[31] H. M. Möller and H. J. Stetter, *Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems*, Numer. Math., 70 (1995), pp. 311–329.
[32] R. E. Moore, *Methods and applications of interval analysis*, SIAM, 1987.
[33] A. P. Morgan, *Polynomial continuation and its relationship to the symbolic reduction of polynomial systems*, Symbolic and Numerical Computation for Artificial Intelligence, Academic Press, New York, 1992, pp. 23–45.
[34] Y. Nakatsukasa, V. Noferini, and A. Townsend, *Computing the common zeros of two bivariate functions via Bézout resultants.* Submitted, 2013.
[35] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research, Springer, second ed., 2006.
[36] P. A. Parrilo, *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*, PhD thesis, California Institute of Technology, 2000.
[37] L. Qi, Z. Wan, and Y.-F. Yang, *Global minimization of normal quartic polynomials based on global descent directions*, SIAM J. Optim., 15 (2006), pp. 275–302.
[38] F. Rouillier, *Solving zero-dimensional systems through the rational univariate representation*,

Appl. Algebra Engrg. Comm. Comput., 9 (1999), pp. 433–461.
[39] F. Rouillier and P. Zimmermann, *Efficient isolation of polynomial real roots*, J. Comput. Appl. Math., 162 (2003), pp. 33–50.
[40] M. Šebek, S. Pejchová, D. Henrion, and H. Kwakernaak, *Numerical methods for zeros and determinant of polynomial matrix*, in Proceedings of the IEEE Mediterranean Symposium on New Directions in Control and Automation, Chania, Crete, Greece, June 1996, pp. 488–491.
[41] R. Seidel and N. Wolpert, *On the exact computation of the topology of real algebraic curves*, in Proceedings of the twenty-first annual symposium on Computational geometry, SCG 2005, New York, NY, USA, 2005, ACM, pp. 107–115.
[42] N. Z. Shor, *Class of global minimum bounds of polynomial functions*, Cybern. and Sys. Anal., 23 (1987), pp. 731–734.
[43] L. Sorber, I. Domanov, M. Van Barel, and L. De Lathauwer, *Exact line and plane search for tensor optimization*, ESAT-SISTA Internal Report 13-02, Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium, 2013.
[44] L. Sorber, M. Van Barel, and L. De Lathauwer, *Unconstrained optimization of real functions in complex variables*, SIAM J. Optim., 22 (2012), pp. 879–898.
[45] ———, *Tensorlab v1.0*. Available online, Feb. 2013.
[46] H. J. Stetter, *Matrix eigenproblems are at the heart of polynomial system solving*, SIGSAM Bull., 30 (1996), pp. 22–25.
[47] A. Townsend and L. N. Trefethen, *An extension of Chebfun to two dimensions*. Submitted, 2013.
[48] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[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] J. Verschelde, J. Verlinden, and R. Cools, *Homotopies exploiting newton polytopes for solving sparse polynomial systems*, SIAM J. Numer. Anal., 31 (1994), pp. 915–930.
[51] R. C. Ward, *Balancing the generalized eigenvalue problem*, SIAM J. Statist. Comput., 2 (1981), pp. 141–152.
[52] J. H. Wilkinson, *The evaluation of the zeros of ill-conditioned polynomials. Part I*, Numer. Math., 1 (1959), pp. 150–166.
[53] ———, *Kronecker's canonical form and the QZ algorithm*, Linear Algebra Appl., 28 (1979), pp. 285–303.
[54] K. Zhou and S. I. Roumeliotis, *Multirobot active target tracking with combinations of relative observations*, IEEE Trans. Robot., 27 (2011), pp. 678–695.