Monografías Matemáticas *García de Galdeano* **41**, 171–185 (2018)

COMPANION MATRICES AND JOINT EIGENVECTORS OF COMMUTING FAMILIES OF MATRICES FOR POLYNOMIAL ZERO FINDING

Tomas Sauer

Abstract. Finding the solutions of polynomial systems of equations is equivalent to finding the eigenvalues of a so-called companion matrix, as long as the system has only finitely many solutions. Algebraically, this means to describe the action of multiplication with a fixed polynomial modulo an ideal. In one variable, this general point of view leads to different explicit matrices whose eigenvalues are the zeros of a given polynomial, in several variables this leads to the problem of finding the joint eigenvectors of a commuting family of matrices. The paper studies both cases and gives algorithms relying only on standard methods from numerical linear algebra.

Keywords: Companion matrix, ideal, commuting family of matrices, joint eigenvector.
AMS classification: 13P15, 65F15.

§1. Introduction

We study some aspects of semialgebraical methods for solving systems of polynomial equations, or equivalently, finding the zeros or common zeros of a polynomial or a finite set of polynomials in one or several variables, respectively. "Semialgebraic" means that the method is derived from algebraic principles, usually computational ideal theory, while the algorithm itself must designed to run as stable as possible in a *floating point* environment where all operations are affected by roundoff errors. In contrast to that, purely algebraic methods work in a *symbolic* environment where all numerical operations are carried out *exactly*, usually by using infinite precision rational numbers, cf. [9].

In this paper, we study techniques based on *companion matrices* which transform the computation of zeros of polynomials into an eigenvalue problem that can be accessed by well–established methods from numerical linear algebra. In a single variable, this is nothing but the classical and well–known approach by means of the *Frobenius companion matrix* of a polynomial. We will derive this method as a special case of describing the operation of multiplication by a given polynomial on normal forms modulo a given ideal. By specializing to different bases for the normal form space, we will recover two classical examples of companion matrices, but also a new companion matrix, for which some numerical experiments show a very interesting behavior.

In the multivariate case, the eigenvalue problem itself becomes more challenging: instead of finding the eigenvalues of a single matrix, we need to compute all eigenvalues and *joint eigenvectors* of a commuting family of $n \times n$ matrices, since the eigenvectors allow us to pick

a proper combination of n joint eigenvalues out of n^s possible combinations. An algorithm for that purpose, given by Möller and Tenberg [18], will be presented, analyzed and modified to work in a numerically more stable way. We will prove the correctness of the modification in Theorem 5 and discuss some possible extensions.

The organization of the paper follows the above storyline. In Section 2, we will sketch the algebraic background and give some example of univariate companion matrices. Numerical experiments with one of those companion matrices will be reported in Section 3 while Section 4 then will deal with the computation of joint eigenvalues or joint diagonalizations of the commuting family.

§2. Companion matrices and zeros of polynomials

Companion matrices or *multiplication tables* are a key ingredient in making the computation of (common) zeros of polynomials numerically accessible. Being a standard method in one variable, cf. [11, p. 348] or [15, p. 147ff], its multivariate counterpart was brought to wider attention by Stetter [22] although it is often attributed to Stickelberger, cf. [3]. The main idea is easily described anyway.

Let $F \subset \mathbb{C}[x] = \mathbb{C}[x_1, \dots, x_s]$ be a finite set of polynomials in s variables with complex coefficients which generates an *ideal*

$$\langle F\rangle = \left\{ \sum_{f\in F} g_f\, f : g_f \in \mathbb{C}[x] \right\}.$$

It is easy to see that $x \in \mathbb{C}^s$ is a solution of the system, i.e., $F(x) = 0$ which means $f(x) = 0$, $f \in F$, if and only if $\langle F\rangle(x) = 0$, that is, $f(x) = 0$, $f \in \langle F\rangle$. In other words, the solution set

$$Z(F) := \{x \in \mathbb{C}^s : F(x) = 0\},$$

which can be seen as the *common zeros* of F or the solution of the polynomial system $F(x) = 0$, depends on the *ideal* $\langle F\rangle$, not of it specific *basis* F. Hilbert's celebrated *Basissatz* tells us that any polynomial ideal can be written as $\langle F\rangle$ for some appropriate finite basis F, but recall that neither is the basis unique nor are the representations with respect to the basis. Indeed, most symbolic methods for polynomial zero finding consist of determining a "better" basis from which the solution can be obtained more easily.

Given F, the algebra of polynomials admits a direct sum decomposition

$$\mathbb{C}[x] = \langle F\rangle \oplus N_F$$

where N_F is the vector space of *normal forms* modulo F. For any $p \in \mathbb{C}[x]$, its associated normal form $n(p)$ can be computed efficiently by *reduction* which is the main concept of Gröbner bases and the generalization of Euclidean division to several variables. For an elementary introduction see [6]. Reduction with respect to a Gröbner basis is provided by all computer algebra systems like `Maple` or `Macaulay2`, numerically more stable methods based on homogeneous orthogonal reduction and *H–bases* have been introduced in [19]. The important point in the context of this paper is the following: given any finite subset $F \subset \mathbb{C}[x]$ and $p \in \mathbb{C}[x]$, the normal form $n(p) \in N_F$ can be computed efficiently in finitely many operations.

In this paper, we only consider *zero dimensional* ideals, which are ideals $\langle F\rangle$ such that $\#Z(F) < \infty$. These ideals can be characterized in terms of their normal form spaces as follows.

Proposition 1. *Let F be a finite set of polynomials and N_F a normal form space for $\langle F\rangle$. Then $\langle F\rangle$ is a zero dimensional ideal if and only if* $\dim N_F < \infty$.

From now on we assume that $\langle F\rangle$ is zero dimensional. Let $L : \mathbb{C}[x] \to N_F$ be the linear projector from that associates to each polynomial its normal form modulo $\langle F\rangle$. If $Lp = Lp'$ then $p - p' \in \langle F\rangle$, hence $p(x) = p'(x)$ for any $x \in Z(F)$. Hence, L *interpolates* at $Z(F)$. We define the polynomials

$$\ell_x := L\left(\prod_{x'\in Z(F)\setminus\{x\}} \frac{(x-x')^T(\cdot - x')}{\|x-x'\|_2^2}\right) \in N_F, \qquad x \in Z(F), \tag{1}$$

which satisfy

$$\ell_x(x') = \delta_{x,x'}, \qquad x, x' \in Z(F). \tag{2}$$

Definition 1. The ideal $\langle F\rangle$ is called *radical* if

$$N_F = \operatorname{span}\{\ell_x : x \in Z(F)\},$$

which means that all zeros in $Z(F)$ are *simple*.

If an ideal is not radical, the *multiplicity* of the zero is a finite dimensional differentiation invariant polynomial subspace, cf. [13, 14]. The occurance of multiplicities can complicate the issue, but it can be assumed that all zeros are simple after a preprocessing step which we will first recall in the univariate case.

Example 1. Let $f(x) = (x-\zeta)^n g(x)$, $g(\zeta) \neq 0$, be a univariate polynomial with an n–fold zero at ζ. Then

$$f'(x) = n(x-\zeta)^{n-1} g(x) + (x-\zeta)^n g'(x) =: (x-\zeta)^{n-1} h(x), \qquad h(\zeta) = n\, g(\zeta) \neq 0,$$

so that $(x-\zeta)^{n-1}$ divides both f and f'. Therefore, $f/\gcd(f, f')$ has the same zeros as f, but only with multiplicity 1.

In "ideal language", Example 1 shows that the associated radical ideal for the *principal ideal* $\langle f\rangle$ is the principal ideal $\langle f/\gcd(f, f')\rangle$. An analogous though more intricate procedure for an arbitrary number of variables can be found in [12, p. 49]. Assuming that we first perform this preprocessing step if necessary, we make the following standing assumption for the rest of this paper.

Assumption 2. *Suppose that $\langle F\rangle$ is a zero dimensional radical ideal, hence*

$$\dim N_F = \#Z(F)$$

and $N_F = \operatorname{span}\{\ell_x : x \in Z(F)\}$.

With Assumption 2 and (2), the *interpolation operator* $L : \mathbb{C}[x] \to N_F$ that computes the normal form can be written as

$$N_F \ni n(p) = Lp = \sum_{x\in Z(F)} p(x)\,\ell_x, \qquad p \in \mathbb{C}[x]. \tag{3}$$

More on polynomial interpolation and its ideal theoretic aspects can be found in [7] and the surveys [8, 20].

With this terminology at hand, we are finally ready to define *companion matrices* or *multiplication tables*. To that end, we note that multiplication by an arbitrary polynomial $q \in \mathbb{C}[x]$ and then interpolating we obtain a mapping from N_F to itself:

$$N_F \ni p \mapsto m_q(p) := L(qp) = \sum_{x \in Z(F)} p(x)q(x)\,\ell_x \tag{4}$$

Since L is a linear operator, it follows that

$$m_q(p+p') = L\left(q(p+p')\right) = L\left(qp + qp'\right) = L(qp) + L(qp') = m_q(p) + m_q(p'),$$

hence m_q is a linear operator that can be represented with respect to a basis B of N_F by matrix

$$M_q := \left(m_{b,b'} : b, b' \in B\right), \qquad m_q(b) = \sum_{b' \in B} m_{b,b'}\, b', \quad b \in B. \tag{5}$$

We observe that the companion matrix M_q depends on the space N_F as well as on the concrete basis B that we have chosen. This principle is as straightforward as flexible, as we will point out by means of some univariate examples next.

Example 2. For $s = 1$ we consider the monic polynomial $f(x) = x^{n+1} + a_n x^n + \cdots + a_0$ for which $N_f = \Pi_n$, the space of all polynomials of degree $\le n$, spanned by the monomials $\{1, x, \ldots, x^n\}$, and $q(x) = x$. Then,

$$m_{(\cdot)}\left((\cdot)^j\right) = L\left((\cdot)^{j+1}\right) = (\cdot)^{j+1}, \qquad j = 0, \ldots, n-1,$$

and

$$m_{(\cdot)}\left((\cdot)^n\right)(x) = L\left((\cdot)^{n+1}\right)(x) = x^{n+1} - f(x) = -\sum_{j=0}^{n} a_j\, x^j,$$

so that the associated multiplication table

$$M_{(\cdot)} = \begin{pmatrix} 0 & & & -a_0 \\ 1 & \ddots & & \vdots \\ & \ddots & 0 & -a_{n-1} \\ & & 1 & -a_n \end{pmatrix} \tag{6}$$

is the well–known *Frobenius companion matrix*, cf. [15].

Example 3. Still Let p_n, $n \in \mathbb{N}_0$, be a system of monic *orthogonal polynomials*, thus satisfying a *three term recurrence* of the form

$$p_n = (\cdot + \beta_n)\, p_{n-1} - \gamma_n\, p_{n-2}, \qquad n \in \mathbb{N}_0, \qquad p_0 = 1,\ p_{-1} = 0, \tag{7}$$

and fix some $n > 0$. Then $N_{p_{n+1}} = \Pi_n$ again and with the basis $\{p_0, \ldots, p_n\}$, the recurrence relation (7) yields that

$$m_{(\cdot)}(p_k) = p_{k+1} - \beta_{k+1} p_k + \gamma_{k+1}\, p_{k-1}, \qquad k = 0, \ldots, n-1,$$

and

$$m_{(\cdot)}(p_n) = -\beta_{n+1}p_n + \gamma_{n+1}p_{n-1},$$

yielding the tridiagonal companion matrix

$$M_{(\cdot)} = \begin{pmatrix} -\beta_1 & \gamma_2 & & & \\ 1 & \beta_2 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -\beta_n & \gamma_{n+1} \\ & & & 1 & -\beta_{n+1} \end{pmatrix} \tag{8}$$

well–known in the computation of quadrature nodes, cf. [10].

The final example is less common, but can be found in [2].

Example 4. Still in $s = 1$, we consider $f(x) = a_{n+1}x^{n+1} + \cdots + a_0$, $a_{n+1} \neq 0$, of degree $n+1$ and arbitrary points $\xi_0, \ldots, \xi_{n+1} \in \mathbb{R}$ which, in particular, do not have to be zeros of f. As basis for $N_f = \Pi_n$ we choose the *Newton fundamental polynomials*

$$B = \left\{ b_j := \prod_{k=0}^{j-1}(\cdot - \xi_k) : j = 0, \ldots, n \right\}$$

and the projector given by *Newton form*

$$Lp = \sum_{j=0}^{n} [\xi_0, \ldots, \xi_j]p \, b_j$$

of the interpolation polynomial where $[\xi_0, \ldots, \xi_j]p$ denotes the *divided difference*. The duality between basis polynomials and divided differences yields that

$$\begin{aligned} m_{(\cdot)}(b_j) = L\Big((\cdot)b_j\Big) &= \sum_{k=0}^{n} [\xi_0, \ldots, \xi_k]\Big((\cdot)b_j\Big)\, b_k \\ &= \sum_{k=0}^{n} [\xi_0, \ldots, \xi_k]\Big((\cdot - \xi_j)b_j\Big)\, b_k + \xi_j \sum_{k=0}^{n} [\xi_0, \ldots, \xi_k]b_j \, b_k \\ &= \sum_{k=0}^{n} [\xi_0, \ldots, \xi_k]b_{j+1}\, b_k + \xi_j \sum_{k=0}^{n} [\xi_0, \ldots, \xi_k]b_j \, b_k = b_{j+1} + \xi_j \, b_j, \qquad j = 0, \ldots, n-1, \end{aligned}$$

and, since $[\xi_0, \ldots, \xi_{n+1}]f = a_{n+1}$ and $((\cdot - \xi_n)\, b_n)\,(\xi_j) = 0$, $j = 0, \ldots, n$,

$$\begin{aligned} m_{(\cdot)}(b_n) &= \xi_n \, b_n + (\cdot - \xi_n)b_n - \frac{1}{[\xi_0, \ldots, \xi_{n+1}]f} f \\ &= \xi_n \, b_n + \sum_{j=0}^{n} \left([\xi_0, \ldots, \xi_k]\,((\cdot - \xi_n)b_n) - \frac{[\xi_0, \ldots, \xi_k]f}{[\xi_0, \ldots, \xi_{n+1}]f} \right) b_k \\ &= \xi_n \, b_n - \sum_{j=0}^{n} \frac{[\xi_0, \ldots, \xi_k]f}{[\xi_0, \ldots, \xi_{n+1}]f}\, b_k. \end{aligned}$$

Consequently, the *Newton companion matrix* is of the form

$$M_{(\cdot)} = \begin{pmatrix} \xi_0 & & & -\frac{[\xi_0]f}{[\xi_0,\dots,\xi_{n+1}]f} \\ 1 & \xi_1 & & -\frac{[\xi_0,\xi_1]f}{[\xi_0,\dots,\xi_{n+1}]f} \\ & \ddots & \ddots & \vdots \\ & & 1 & \xi_n - \frac{[\xi_0,\dots,\xi_n]f}{[\xi_0,\dots,\xi_{n+1}]f} \end{pmatrix}, \tag{9}$$

and depends only on the values $f(\xi_0), \dots, f(\xi_{n+1})$, quite is in the spirit of *algebra from values* as introduced, for example, in [23].

So far, we have considered only univariate examples. For $s > 1$, the fundamental multiplication tables are those of the form $M_{(\cdot)_j}$ which correspond to multiplication with coordinate polynomials. Since

$$m_{qq'}(p) = L(qq'p) = L\left(q\, L(q'p)\right) = L\left(q\, m_{q'}(p)\right) = m_q\left(m_{q'}(p)\right), \qquad p \in N_F,$$

and

$$m_{q+q'}(p) = L\left((q+q')p\right) = L(qp) + L(q'p) = m_q(p) + m_{q'}(p), \qquad p \in N_F,$$

we can immediately draw the following conclusions.

Theorem 3. *The matrices $M_j := M_{(\cdot)_j}$, $j = 1, \dots, s$, commute and generate the algebra of multiplication tables, i.e., $M_j M_k = M_k M_j$, $j,k = 1, \dots, s$, and*

$$M_q = q(M) = \sum_{\alpha \in \mathbb{N}_0^s} q_\alpha\, M^\alpha = \sum_{\alpha \in \mathbb{N}_0^s} q_\alpha\, M_1^{\alpha_1} \cdots M_s^{\alpha_s}, \qquad q = \sum_{\alpha \in \mathbb{N}_0^s} q_\alpha\, (\cdot)^\alpha. \tag{10}$$

Example 5. Suppose that F is such that $N_F = \Pi_n$ with the monomial basis $B = \{(\cdot)^\alpha : |\alpha| \le n\}$ in standard multiindex notation. In this case, there exist a basis of $\langle F \rangle$ which is of the form $G := \{g_\alpha := (\cdot)^\alpha + \tilde{g}_\alpha : \tilde{g}_\alpha \in \Pi_n,\ |\alpha| = n+1\}$, and this basis is even an *H–basis*. Then

$$m_{(\cdot)_j}\left((\cdot)^\alpha\right) = (\cdot)^{\alpha+\epsilon_j}, \qquad |\alpha| \le n,$$

where $\epsilon_j \in \mathbb{N}_0^s$ denotes the jth coordinate multiindex, and

$$m_{(\cdot)_j}\left((\cdot)^\alpha\right) = -\tilde{g}_{\alpha+\epsilon_j} = -\sum_{|\beta| \le n} \tilde{g}_{\alpha+\epsilon_j,\beta}\, (\cdot)^\beta, \qquad |\alpha| = n.$$

Consequently, arranging these relations into blocks according to the total degree, the jth companion matrix has the block Hessenberg form

$$M_j = \begin{pmatrix} 0 & & & G_0 \\ S_1 & & & G_1 \\ & \ddots & & \vdots \\ & & S_n & G_n \end{pmatrix}, \tag{11}$$

where

$$S_k = \left(\delta_{\alpha,\beta+\epsilon_j} : \begin{matrix} |\alpha| = k \\ |\beta| = k-1 \end{matrix}\right) \in \mathbb{C}^{\binom{k+s}{s-1} \times \binom{k-1+s}{s-1}}, \qquad k = 1, \dots, n,$$

and

$$G_k = \left(-\tilde{g}_{\beta+\epsilon_j,\alpha} : \begin{array}{l} |\alpha| = k \\ |\beta| = n \end{array} \right) \in \mathbb{C}^{\binom{k+s}{s-1} \times \binom{n+s}{s-1}}, \qquad k = 0, \ldots, n.$$

In this respect, (11) is the exact block analogy of the Frobenius companion matrix (6) from Example 2 and can be directly read off the coefficients of the *monic* Gröbner or H–basis of the ideal $\langle F \rangle$.

The practical use of companion matrices lies in the following theorem which has different attributions and an extremely simple proof.

Theorem 4 (Stetter, Sticklberger)**.** *If $\langle F \rangle$ is zero dimensional and radical, then the eigenvalues of M_j are x_j, $x \in Z(F)$, and the associated eigenvectors are ℓ_x, $x \in Z(F)$.*

Proof. Note that by (2)

$$m_{(\cdot)_j}(\ell_x) = L\Big((\cdot)_j \ell_x\Big) = \sum_{x' \in Z(F)} x'_j \ell_x(x')\, \ell_{x'} = x_j\, \ell_x \tag{12}$$

holds for any $x \in Z(F)$, hence the coefficient vectors of the ℓ_x are the $\#Z(F)$ linearly independent joint eigenvectors of the $\#Z(F) \times \#Z(F)$-matrices M_j. □

Remark 1. Note that the eigenvectors ℓ_x do *not* depend on the matrix M_j. Indeed, we even have that

$$M_q \ell_x = q(M)\ell_x = \sum_{\alpha \in \mathbb{N}_0^s} q_\alpha M^\alpha\, \ell_x = \sum_{\alpha \in \mathbb{N}_0^s} q_\alpha x^\alpha\, \ell_x = q(x)\, \ell_x$$

for any polynomial $q \in \mathbb{C}[x]$.

Remark 2. For $s = 1$, Theorem 4 follows directly from inspecting the Newton Companion matrix from Example 4: if the parameters $\xi_0, \ldots, \xi_n$ are, by accident, the zeros of f, then $[\xi_0, \ldots, \xi_j]f = 0$, $j = 0, \ldots, n$, and the eigenvalues of the matrix are the zeros of the polynomial. Since all companion matrices only differ by a change of basis, i.e., a similarity transform, the same also holds for the companion matrices of the other examples. On the other hand, (12) is nothing but computing the diagonal companion matrix with respect to the basis $B = \{\ell_x : x \in Z(F)\}$.

§3. A numerical experiment

Returning to $s = 1$, we briefly perform some numerical experiments concerning the Newton companion matrix and its iterative application. For simplicity, we assume that f is a *monic* polynomial and start with some arbitrary, for example random, choice

$$\xi^0 = \left(\xi_0^0, \ldots, \xi_n^0\right),$$

from which compute a sequence of companion matrices

$$M_j := \begin{pmatrix} \xi_0^j & & & -[\xi_0^j]f \\ 1 & \xi_1^j & & -[\xi_0^j, \xi_1^j]f \\ & \ddots & \ddots & \vdots \\ & & 1 & \xi_n^j - [\xi_0^j, \ldots, \xi_n^j]f \end{pmatrix}, \qquad j \in \mathbb{N},$$

	1 iteration		2 iterations		3 iterations	
n	avg	max	avg	max	avg	max
2	1.2726e-14	1.1130e-11	1.4280e-17	1.5266e-16	1.0138e-17	1.5266e-16
5	2.6031e-11	2.7078e-08	1.5354e-16	7.7716e-15	1.4655e-16	3.8858e-15
8	1.0356e-07	2.3336e-04	2.8920e-16	4.7851e-14	2.3223e-16	5.6621e-15
10	9.2646e-06	3.0206e-02	2.4375e-09	8.2716e-06	4.8751e-09	1.6543e-05
15	2.2419e-04	1.1340e-01	6.0210e-04	2.7328e-01	6.9069e-04	2.9682e-01
20	1.6115e-03	7.4816e-01	6.0930e-03	6.2219e-01	7.0142e-03	6.6549e-01

Table 1: Random polynomials and their zeros, average and maximal errors over 1000 trials.

	# of iterations				
n	1	2	3	4	5
5	13.900	15.913	15.930	15.969	16.029
10	11.034	15.243	15.414	15.428	15.425
15	7.9267	12.4128	13.6033	13.6274	13.6394
20	5.3069	7.4589	8.9817	9.0395	9.0391
30	1.79008	1.00444	0.95683	0.91752	0.87821

Table 2: Logarithmic average, number of correct digits in average over 1000 trials

by choosing ξ^{j+1} as the eigenvalues of M_j. The experiment proceeds in the following way: we first compute n random points $\zeta_j \in [0,1]$ and set $f(x) = \prod_{j=1}^{n}(x-\zeta_j)$. Moreover, we choose a random initialization for $\xi^0 \in [0,1]^{n+1}$ and then run, with the same setup, several iterations of of the Newton companion process. This experience is repeated 1000 times, of course with new random points, and we average the absolute value of the errors over all zeros and all trials. In addition, we also record the maximal deviation.

The results are listed in Table 1. For small degree polynomials we can record a significant improvement in accuracy, while for degrees larger than 10, there iterations do not give better results. The reason is that the numerical inaccuracies in the computation of the divided difference result in eigenvectors with a significant imaginary part and resubsituting these values into the process leads even to increased errors in some cases. Moreover, Table 1 is, however, somewhat misleading as it only determines the arithmetic mean of all errors and therefore a single case with an error of 10^{-3} dominates many cases with an error of 10^{-10}.

Indeed, iterative application of Newton companion matrices can often lead to a significant improvement, at least for moderate degrees, as Table 2 shows, where the average of the logarithmic error is listed, i.e., the average number of *correct decimal digits* in the solutions. At least up to degree $n = 20$, the iterations improve the numerical quality significantly. Of course, further investigation and quantitative statements would be needed here which is currently under investigation.

§4. Joint eigenvectors

Now we turn our attention to $s > 1$, the "truly" multivariate situation. Theorem 4 tells us that, numerically, the problem of finding the common zeros $Z(F)$ of $F \subset \mathbb{C}[x]$ is equivalent

to finding the joint eigenvalues of the commuting family M_j. We will not focus here on the question how these matrices are obtained; this can be done by Gröbner or H–basis methods, see, for example [1, 17] or [16], where also some numerical issues concerning the choice of basis are addressed. Moreover, a method to obtain companion matrices for polynomials whose coefficients are given implicitly as kernels of certain matrices by fast methods from numerical linear algebra has been pointed out in [21] and was the starting point for these investigations.

Möller and Tenberg [18] gave an algorithm to compute joint eigenvectors and the associated eigenvalues based on intersections on eigenspaces, while an adapted *QR* method to compute joint eigenvectors and associated eigenvalues by decomposing the companion matrices *simultaneously* has been addressed in [4, 5]. Here, we will reconsider the eigenspaces intersection method, first recalling the Möller–Tenberg algorithm and then introducing a modified version that is more suitable for numerical computations.

Specifically, we want to solve the following apparently innocent problem from numerical linear algebra.

Problem 1. Given s commuting matrices $M_1, \ldots, M_s \in \mathbb{C}^{n\times n}$, find a matrix $V \in \mathbb{C}^{n\times n}$ whose columns are the n linearly joint independent eigenvectors of M_j, $j = 1, \ldots, s$.

Remark 3. Before stating the algorithms, let us recall some basic facts first:

1. For general commuting families, Problem 1 as stated here cannot be solved because there does not have to exist a basis of eigenvectors for the M_j if some of these bases have multiple eigenvalues where geometric and algebraic multiplicity do not match. In our particular case here, however, Assumption 2 ensures that the coefficient vectors of the ℓ_x, $x \in Z(F)$, are the eigenvectors we are looking for. And (2) immediately guarantees that the ℓ_x are linearly independent: suppose that a coefficient vector $(a_x : x \in X)$ satisfies
$$0 = \sum_{x\in Z(F)} a_x \ell_x =: p.$$
Then evaluation of p on $Z(F)$ implies that $a_x = 0$, $x \in Z(F)$.

2. Once the matrix V is determined, it simultaneously diagonalizes the matrices M_j,
$$V^{-1} M_j V = D_j = \begin{pmatrix} \lambda_{j1} & & \\ & \ddots & \\ & & \lambda_{jn} \end{pmatrix}, \qquad j = 1, \ldots, s,$$
and we get
$$Z(F) = \{(\lambda_{1k}, \ldots, \lambda_{sk}) : k = 1, \ldots, n\}. \tag{13}$$
This highlights the importance of the joint *eigenvectors*: they connect the eigenvalues to give the n elements of $Z(F)$.

3. We rely on a function $\Lambda : \mathbb{C}^{n\times n} \to \mathbb{C}^n \times \mathbb{C}^{n\times n}$, such that, given a square matrix $A \in \mathbb{C}^{n\times n}$, $\Lambda(A) = (\lambda, E)$ computes a vector $\lambda \in \mathbb{C}^n$ of eigenvalues and a matrix $E \in \mathbb{C}^{n\times n}$ that contains a normalized basis of eigenvectors, provided the matrix can be diagonalized. Such a function is provided by `Matlab` and `octave` under the name `eig`.

Definition 2. We write a vector $\lambda \in \mathbb{C}^n$ by *multiplicities* as $(\hat\lambda, \mu)$, $\hat\lambda \in \mathbb{C}^m$, $\mu \in \mathbb{N}^m$, where $\hat\lambda_j \neq \hat\lambda_k$, $1 \le j < k \le m$, and

$$\lambda = (\underbrace{\hat\lambda_1, \dots, \hat\lambda_1}_{\mu_1}, \dots, \underbrace{\hat\lambda_m, \dots, \hat\lambda_m}_{\mu_m}), \qquad \sum_{j=1}^m \mu_j = n.$$

Both methods to be stated now, Algorithm 1 and Algorithm 2, can best be described by means of column partitioned $n \times n$ matrices which we will write as

$$V = [V_1 \mid \dots \mid V_\ell], \qquad V_j \in \mathbb{C}^{n \times n_j}, \qquad j = 1, \dots, \ell,$$

where $\ell = \ell(V)$ stands for the number of column blocks. With this terminology, we can state the first method which computes the joint eigenvectors by successively refining eigenspaces.

Algorithm 1 (Möller & Tenberg, [18])**.**
Given: $M_1, \dots, M_s \in \mathbb{C}^{n \times n}$.

1. Compute $(\lambda, E) = \Lambda(M_1)$, $\lambda = (\hat\lambda, \mu)$, let V be a column permutation of E such that

   $$V = [V_1 \mid \dots \mid V_m], \qquad M_j V_k = \hat\lambda_k V_k, \quad k = 1, \dots, m,$$

   and set $\ell(V) = m$.
2. For $j = 2, \dots, s$
   (a) For $k = 1, \dots, \ell := \ell(V)$
      i. Compute, by means of the *pseudoinverse* V_k^+,

      $$(\lambda, E) = \Lambda(V_k^+ M_j V_k), \qquad \lambda = (\hat\lambda, \mu). \tag{14}$$

      ii. Partition $E = [E_1 \mid \dots \mid E_m]$ according to $\hat\lambda$ and μ.
      iii. Replace V_k by $[V_k E_1 \mid \dots \mid V_k E_m]$, i.e., $\ell(V_k) = m$.
   (b) Replace $\ell(V)$ by $\ell(V_1) + \dots + \ell(V_\ell)$.

Result: matrix V such that $V^{-1} M_j V = D_j$, where D_j is diagonal.

Remark 4. Algorithm 1 also works in the case of multiple zeros where the matrices M_j do not have a basis of eigenvalues. For details see [18].

The second algorithm is a variation of the Möller–Tenberg method that uses explicit subspace intersection. To that end, we identify a matrix $A \in \mathbb{C}^{n \times a}$ with the subspace $A\mathbb{C}^a \subset \mathbb{C}^n$ spanned by the columns of A. In this sense, $A \cap B$, $A \in \mathbb{C}^{n \times a}$, $B \in \mathbb{C}^{n \times b}$, stands for a matrix whose columns span the intersection $A\mathbb{C}^a \cap B\mathbb{C}^b$. How to compute such an intersection will be recalled in Algorithm 3. If two spaces have only trivial intersection, i.e., $A \cap B = \{0\}$, then $A \cap B$ is represented by the empty matrix as in `Matlab`.

Algorithm 2 (Eigenspace intersection)**.**
Given: $M_1, \dots, M_s \in \mathbb{C}^{n \times n}$.

1. Compute $(\lambda, E) = \Lambda(M_1)$, $\lambda = (\hat\lambda, \mu)$, let V be a column permutation of E such that

   $$V = [V_1 \mid \dots \mid V_m], \qquad M_j V_k = \hat\lambda_k V_k, \quad k = 1, \dots, m,$$

   and set $\ell(V) = m$.

2. For $j = 2, \ldots, s$
   (a) Compute
   $$(\lambda, E) = \Lambda(V^{-1} M_j V), \qquad \lambda = (\hat{\lambda}, \mu), \tag{15}$$
   and partition $E = [E_1 \mid \ldots \mid E_m]$ according to $\hat{\lambda}$ and μ.
   (b) For $k = 1, \ldots, \ell = \ell(V)$
      i. Replace V_k by $[V_k \cap VE_1 \mid \ldots \mid V_k \cap VE_m]$.
      ii. Set $\ell(V_k) = \#\{t : \dim(V_k \cap VE_t) \geq 1\}$.
3. Replace $\ell(V)$ by $\ell(V_1) + \cdots + \ell(V_\ell)$.

Result: matrix V such that $V^{-1} M_j V = D_j$, where D_j is diagonal.

Theorem 5. *If the matrices M_j, $j = 1, \ldots, s$, are companion matrices of a zero dimensional radical ideal, then Algorithm 2 computes a matrix V that simultaneously diagonalizes the matrices M_j, $j = 1, \ldots, s$.*

Proof. The principle of Algorithm 2 is that after j steps the blocks in the partitioning

$$V = [V_1 \mid \ldots \mid V_\ell]$$

are exactly the nontrivial intersections of eigenspaces of $M_1, \ldots, M_j$. We prove this fact by induction on j where the case $j = 1$ is a trivial consequence of the first step of the algorithm. Suppose that the claim is valid for some $j \geq 1$. Since

$$M_{j+1} VE_t = VV^{-1} M_{j+1} VE_t = \hat{\lambda}_t \, VE_t, \qquad t = 1, \ldots, m,$$

any intersection $V_k \cap VE_t$ is a joint eigenspace of M_{j+1} and one of $M_1, \ldots, M_j$ by the induction hypothesis, thus of $M_1, \ldots, M_{j+1}$. Moreover, E and V are nonsingular, hence $\mathbb{C}^n = VE_1 \oplus \cdots \oplus VE_m$ and therefore

$$V_k = V_k \cap \mathbb{C}^n = V_k \cap \left(\bigoplus_{t=1}^{m} VE_t \right) = \bigoplus_{t=1}^{m} (V_k \cap VE_t),$$

which yields $\mathbb{C}^n = V_1 \oplus \cdots \oplus V_\ell$ at each level and proves that V contains *all* intersections. Since the intersections of eigenspaces of $M_1, \ldots, M_s$ are of dimensions 0 or 1, the case $j = s$ proves the claim. □

Next, we collect some observations on the two algorithms.

Remark 5. A first view, Algorithm 1 and Algorithm 2 appear to be some "overkill". Indeed, if there is *a single* matrix M_j with only simple eigenvalues, then its decomposition $(\lambda, E) = \Lambda(M_j)$ already gives a matrix E that diagonalizes *all* the M_j. Since, moreover, multiple eigenvalues are very unlikely after small perturbations of the matrix, one may ask why all this effort should be necessary. The answer is simple: in polynomial systems it quite often happens that the solutions lie on a grid, i.e., belong to a subset of $X_1 \times \cdots \times X_s$, for finite sets $X_j \subset \mathbb{C}$, and then M_j has just $\#X_j$ different eigenvalues, each of them with respective multiplicities. Even if they may all different after small perturbations in the computation of the companion matrices, the eigenspace structure is still necessary to find the proper combinations of these values.

Remark 6. This approach also connects to projection methods. Indeed, if $p \in \Pi_1$ is a linear polynomial such that $p(x) \neq p(x')$, $x, x' \in Z(F)$ – and *almost any* linear function has this property – then

$$M_p = p_0 I + \sum_{j=1}^{s} p_j M_j$$

has n simple eigenvalues and the eigenvectors are the joint eigenvectors of any M_q, $q \in \mathbb{C}[x]$, according to Theorem 3. Hence, the matrix V with these eigenvectors as columns diagonalizes all M_j, $j = 1, \dots, s$. The main problem, however, is to find an *a priori* p such that the eigenvalues of M_p are sufficiently well separated to give a numerically meaningful result.

Remark 7. Since in both algorithms nothing has to be done in step 2a) and 2b), respectively, as soon as $\ell(V_k) = 1$, both algorithms can be terminated as soon as $\ell(V) = n$. If all eigenvalues of M_1 are simple, this even happens after the first step.

Remark 8. While the Möller–Tenberg method in Algorithm 1 is faster since it has to solve smaller eigenvalue problems in (14) compared to those in (15), it is more sensitive to the order of the matrices and to small perturbations in the companion matrices than Algorithm 2 since it only refines the spaces and cannot compensate small errors made in previous steps.

Remark 9. In both algorithms it is reasonable to normalize the columns of V after each iteration. It is not included in the descriptions since it is not needed for correctness but it usually stabilizes the computations significantly.

Remark 10. The *prediagonalisation* in (15), using $V^{-1}M_jV$ would not be necessary for the method to work, but it improves the performance of eigenvalue computations. Whenever a column of V is already an eigenvalue of M_j, then we get that

$$V^{-1}M_jV = \begin{pmatrix} * & \dots & * & 0 & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & 0 & * & \dots & * \\ 0 & \dots & 0 & \lambda & 0 & \dots & 0 \\ * & \dots & * & 0 & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & \dots & * & 0 & * & \dots & * \end{pmatrix},$$

where λ is the associated eigenvalue. Thus, the eigenvalue routine does only have to do reductions where needed.

For the sake of completeness, we finally recall an easy and numerically stable way to compute the intersection of two subspaces.

Algorithm 3 (Subspace intersection)**.**
Given: $A \in \mathbb{C}^{n\times a}$, $B \in \mathbb{C}^{n\times b}$ of rank a and b, respectively, $a, b \leq n$.

1. Compute a singular value decomposition

$$\begin{pmatrix} A \mid -B \end{pmatrix} = U\Sigma V^*, \qquad U \in \mathbb{C}^{n\times n},\ \Sigma \in \mathbb{C}^{n\times(a+b)},\ V \in \mathbb{C}^{(a+b)\times(a+b)},$$

with singular values $\sigma_1 \geq \cdots \sigma_k > \sigma_{k+1} = \cdots = \sigma_{\min(n,a+b)} = 0$.

2. Set $C = AV_{1:a,k+1:a+b}$.

Result: $C = A \cap B$.

The validity of this algorithm is easily verified: a vector $0 \neq w \in \mathbb{C}^n$ belongs to $A \cap B$ if and only if there are nonzero vectors $u \in \mathbb{C}^a$ and $v \in \mathbb{C}^b$ such that $w = Au = Bv$, hence

$$\begin{pmatrix} A \mid -B \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = 0$$

so that Au belongs to the kernel if and only if u consists of the first a columns of a nontrivial element of the kernel of $\begin{pmatrix} A \mid -B \end{pmatrix}$, and the rank assumption ensures that $u \neq 0$.

In numerical practice, Algorithm 3 works by thresholding the singular values with respect to a given tolerance $\tau > 0$, setting all singular values with $\sigma_j \leq \tau$ to zero. Moreover, the version given here always computes C as a subspace of A. An alternative way could be the symmetric choice

$$C = \frac{1}{2} \left(AV_{1:a,k+1:a+b} + BV_{a+1:a+b,k+1:a+b} \right)$$

which better compensates small errors in A and B by averaging.

The (implicit) thresholding in Algorithm 3 also suggests an important improvement to Algorithm 2. Note that if the threshold τ is too small, some subspace intersections will be missed while a too large threshold generates false intersections. This can be fixed to some extend by adapting the threshold in such a way that the intersections at least span a space of proper dimension, as done in the following procedure.

Algorithm 4 (Adaptive threshold control)**.**

1. Fix $\rho > 1$.
2. Compute $W = V[E_1 \mid \ldots \mid E_m] = [W_1 \mid \ldots \mid W_m]$.
3. Repeat
   (a) Compute

$$X_t := V_k \cap W_t, \quad t = 1, \ldots, m \qquad \text{and} \qquad \ell' = \sum_{t=1}^{m} \dim X_t$$

   by Algorithm 3 based on the threshold τ.
   (b) If $\ell' < m$ replace τ by $\rho\tau$.
   (c) If $\ell' > m$ replace τ by $\rho^{-1}\tau$.

   until $\ell' = m$.

This algorithm terminates. This is due to the fact ℓ' is monotonically increasing with respect to τ and the extremal values $\tau = 0$ and $\tau = \sigma_1$ give smaller and larger intersections, respectively.

Acknowledgements

I am grateful to H. Michael Möller and Juan Manuel Peña for the long time collaboration on these issues and many fruitful discussions. Moreover, I want to thank my Bachelor student Tobias Fuchs whose thoughtful questions triggered the development of Algorithm 2.

References

[1] Auzinger, W., and Stetter, H. J. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Numerical mathematics, Singapore 1988*, vol. 86 of *Internat. Schriftenreihe Numer. Math.* Birkhäuser, Basel, 1988, pp. 11–30.

[2] Calvetti, D., Reichel, L., and Sgallari, F. A modified companion matrix method based on Newton polynomials. In *Fast Algorithms for Structured Matrices: Theory and Applications,*, V. Olshevsky, Ed., vol. 323 of *Contemporary Mathematics*. Amer. Math. Soc., 2003.

[3] Cohen, A. M., Cuypers, H., and Sterk, M., Eds. *Some Tapas of Computer Algebra*, vol. 4 of *Algorithms and Computations in Mathematics*. Springer, 1999.

[4] Cortés, V., Peña, J. M., and Sauer, T. Simultaneous triangularization of commuting matrices for the solution of polynomial equations. *Cent. Eur. J. Math 10* (2012), 277–291.

[5] Cortés, V., Peña, J. M., and Sauer, T. Simultaneous triangulation of commuting families of matrices—why and how precisely? In *Eleventh International Conference Zaragoza-Pau on Applied Mathematics and Statistics*, vol. 37 of *Monogr. Mat. García Galdeano*. Prensas Univ. Zaragoza, Zaragoza, 2012, pp. 103–113.

[6] Cox, D., Little, J., and O'Shea, D. *Ideals, Varieties and Algorithms*, 2. ed. Undergraduate Texts in Mathematics. Springer–Verlag, 1996.

[7] de Boor, C. Ideal interpolation. In *Approximation Theory XI, Gaitlinburg 2004* (2005), C. K. Chui, M. Neamtu, and L. L. Schumaker, Eds., Nashboro Press, pp. 59–91.

[8] Gasca, M., and Sauer, T. Polynomial interpolation in several variables. *Advances Comput. Math. 12* (2000), 377–410.

[9] Gathen, J. v. z., and Gerhard, J. *Modern Computer Algebra*. Cambridge University Press, 1999.

[10] Gautschi, W. *Numerical Analysis. An Introduction*. Birkhäuser, 1997.

[11] Golub, G., and van Loan, C. F. *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, 1996.

[12] González-Vega, L., Rouillier, F., Roy, M.-F., and Trujillo, G. Symbolic recipes for real solution. In *Some Tapas in Computer Algebra*, A. M. Cohen, H. Cuypers, and M. Sterk, Eds., vol. 4 of *Algorithms and Computations in Mathematics*. Springer, 1999, ch. 2, pp. 121–162.

[13] Gröbner, W. Über das Macaulaysche inverse System und dessen Bedeutung für die Theorie der linearen Differentialgleichungen mit konstanten Koeffizienten. *Abh. Math. Sem. Hamburg 12* (1937), 127–132.

[14] Gröbner, W. Über die algebraischen Eigenschaften der Integrale von linearen Differentialgleichungen mit konstanten Koeffizienten. *Monatsh. Math. 47* (1939), 247–284.

[15] Householder, A. S. *The Theory of Matrices in Numerical Analysis*. Blaisdell Publishing Company, 1964. Dover reprint 2006.

[16] Möller, H. M., and Sauer, T. H–bases II: Applications to numerical problems. In *Curve and Surface fitting: Saint–Malo 1999* (2000), A. Cohen, C. Rabut, and L. L. Schumaker, Eds., Vanderbilt University Press, pp. 333–342.

[17] Möller, H. M., and Stetter, H. J. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numer. Math. 70* (1995), 311–329.

[18] Möller, H. M., and Tenberg, R. Multivariate polynomial system solving using intersections of eigenspaces. *J. Symbolic Comput. 32* (2001), 513–531.

[19] Sauer, T. Gröbner bases, H–bases and interpolation. *Trans. Amer. Math. Soc. 353* (2001), 2293–2308.

[20] Sauer, T. Polynomial interpolation in several variables: Lattices, differences, and ideals. In *Multivariate Approximation and Interpolation*, M. Buhmann, W. Hausmann, K. Jetter, W. Schaback, and J. Stöckler, Eds. Elsevier, 2006, pp. 189–228.

[21] Sauer, T. Prony's method in several variables. *Numer. Math. 136* (2017), 411–438. arXiv:1602.02352. doi:10.1007/s00211-016-0844-8.

[22] Stetter, H. J. Matrix eigenproblems at the heart of polynomial system solving. *SIGSAM Bull. 30*, 4 (1995), 22–25.

[23] Stetter, H. J. *Numerical Polynomial Algebra*. SIAM, 2005.

Tomas Sauer
Lehrstuhl für Mathematik mit Schwerpunkt Digitale Bildverarbeitung
FORWISS
Innstr. 43
D–94032 Passau, Germany
Tomas.Sauer@uni-passau.de