Polynomial Roots from Companion Matrix Eigenvalues

Alan Edelman*
H. Murakami†

January 1, 1994

Abstract

In classical linear algebra, the eigenvalues of a matrix are sometimes defined as the roots of the characteristic polynomial. An algorithm to compute the roots of a polynomial by computing the eigenvalues of the corresponding companion matrix turns the tables on the usual definition. We derive a first order error analysis of this algorithm that sheds light on both the underlying geometry of the problem as well as the numerical error of the algorithm. Our error analysis expands on work by Van Dooren and Dewilde in that it states that the algorithm is backwards normwise stable in a sense that must be defined carefully. Regarding the stronger concept of a small componentwise backwards error, our analysis predicts a small such error on a test suite of eight random polynomials suggested by Toh and Trefethen. However, we construct examples for which a small componentwise relative backwards error is neither predicted nor obtained in practice. We extend our results to polynomial matrices, where the result is essentially the same, but the geometry becomes more complicated.

1 Introduction

Computing roots of polynomials may be posed as an eigenvalue problem by forming the companion matrix. The eigenvalues of this matrix may be found by computing the eigenvalues of this nonsymmetric matrix using standard versions of balancing (very important for accuracy!!) [7] followed by the QR algorithm as may be found in LAPACK or its precursor EISPACK. This is how the MATLAB command `roots` performs its computation.

As Cleve Moler has pointed out in [6], this method may not be the best possible because

> it uses order n^2 storage and order n^3 time. An algorithm designed specifically for polynomial roots might use order n storage and n^2 time. And the roundoff errors introduced correspond to perturbations in the elements of the companion matrix, not the polynomial coefficients.

Moler continues by pointing out that

> We don't know of any cases where the computed roots are not the exact roots of a slightly perturbed polynomial, but such cases might exist.

This paper investigates whether such cases might exist. Let $\hat{r}_i$, $i = 1, \ldots, n$ denote the roots of $p(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} + x^n$ that are computed by this method. Further assume that the $\hat{r}_i$ are the exact roots of

$$\hat{p}(x) = \hat{a}_0 + \hat{a}_1 x + \ldots + \hat{a}_{n-1}x^{n-1} + x^n.$$

*Department of Mathematics Room 2-380, Massachusetts Institute of Technology, Cambridge, MA 02139, `edelman@math.mit.edu`. Supported by NSF grant DMS-9120852 and the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

†Quantum Chemistry Lab., Department of Chemistry, Hokkaido University, Sapporo 060, Japan, `hiroshi@chem2.hokudai.ac.jp`.

What does it mean to say that $\hat{p}$ is a slight perturbation of p? We give four answers, the best one is saved for last. In the definitions, $O(\epsilon)$ is meant to signify a small though unspecified multiple of machine precision.

1. The "calculus" definition would require that first order perturbations of the matrix lead to first order perturbations of the coefficients.

2. A *normwise* answer that is compatible with standard eigenvalue backward error analyses is to require that
$$\max |a_i - \hat{a}_i| = O(\epsilon)||C||,$$
where C denotes the companion matrix corresponding to p, and ϵ denotes the machine precision.

3. A second normwise answer that would be even better is to require that
$$\max |a_i - \hat{a}_i| = O(\epsilon)||\text{balance}(C)||.$$
Standard algorithms for eigenvalue computations balance a matrix C by finding a diagonal matrix T such that $B = T^{-1}CT$ has a smaller norm than C.

4. The strongest requirement should be that
$$\max \frac{|a_i - \hat{a}_i|}{|a_i|} = O(\epsilon).$$
This is what we will mean by a small *componentwise perturbation.* If $a_i = 0$, then one often wants $\hat{a}_i$ to be 0 too, i.e., ideally one preserves the sparsity structure of the problem.

It was already shown by Van Dooren and Dewilde [11, p.576] by a block Gaussian elimination argument that the calculus definition holds. Their argument is valid in the more complicated case of polynomial matrices. It immediately follows that good normwise answers are available, though it is not clear what exactly are the constants involved. We found that integrating work by Arnold [1] with Newton-like identities allows for an illuminating geometrical derivation of a backward error bound that is precise to first order. Our work improves on [11] in that we derive the exact first order perturbation expression which we test against numerical experiments. Numerical experiments by Toh and Trefethen [9] compare this algorithm with the Jenkins-Traub or the Madsen-Reid algorithm. These experiments indicate that all three algorithms have roughly similar stability properties, and further that there appears to be a close link between the pseudospectra of the balanced companion matrix and the pseudozero sets of the corresponding polynomial.

2 Error Analysis

2.1 Problem Statement and Solution

Let $p(z) = a_0 + a_1 z + \ldots + a_{n-1} z^{n-1} + z^n$ be any monic polynomial. The companion matrix of this polynomial,
$$C = \begin{pmatrix} 0 & & & & -a_0 \\ 1 & 0 & & & -a_1 \\ & 1 & 0 & & -a_2 \\ & & \ddots & & \vdots \\ & & & 1 & -a_{n-1} \end{pmatrix} \tag{1}$$
has characteristic polynomial $P_C(z) \equiv \det(zI - C) = p(z)$.

If E is a dense perturbation matrix with "small" entries, the natural error analysis question is the computation of $P_{C+E}(z) - P_C(z) \equiv \delta a_0 + \delta a_1 z + \ldots + \delta a_{n-1} z^{n-1}$. In MATLAB style notation, we are studying $\mathtt{poly}(C+E) - \mathtt{poly}(C)$ to first order.

Our result is

Theorem 2.1 *To first order, the coefficient of z^{k-1} in $P_{C+E}(z) - P_C(z)$ is*

$$\sum_{m=0}^{k-1} a_m \sum_{i=k+1}^{n} E_{i,i+m-k} - \sum_{m=k}^{n} a_m \sum_{i=1}^{k} E_{i,i+m-k}, \tag{2}$$

where a_n is defined to be 1.

In particular, we see that a small perturbation E introduces errors in the coefficients that are linear in the E_{ij}. Since it is well known that standard eigenvalue procedures compute eigenvalues of matrices with a "small" backward error, we have a precise sense in which we claim that there is a polynomial near $P_C(z)$ whose exact roots are computed in this manner.

For convenience, we state this result in a matrix times vector format. Let

$$\begin{array}{l} f_{k,d} \equiv \sum_{i=1}^{k} E_{i,i+d} \text{ and} \\ b_{k,d} \equiv \sum_{i=k}^{n} E_{i,i+d}. \end{array}$$

These are the forward and backward "plus-scans" or "prefixes" of the dth diagonal of E. Our result is that the matrix-vector product

$$\begin{pmatrix} \delta a_0 \\ \delta a_1 \\ \delta a_2 \\ \vdots \\ \delta a_{n-1} \end{pmatrix} = \begin{pmatrix} b_{2,-1} & -f_{1,0} & -f_{1,1} & \ldots & -f_{1,n-3} & -f_{1,n-2} & -f_{1,n-1} \\ b_{3,-2} & b_{3,-1} & -f_{2,0} & \ldots & -f_{2,n-4} & -f_{2,n-3} & -f_{2,n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ b_{n,-(n-1)} & b_{n,-(n-2)} & b_{n,-(n-3)} & \ldots & b_{n,-1} & -f_{n-1,0} & -f_{n-1,1} \\ 0 & 0 & 0 & \ldots & 0 & 0 & -f_{n,0} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n = 1 \end{pmatrix} \tag{3}$$

is correct to first order. The $n \times (n+1)$ matrix above contains backward prefixes in the lower triangular part and forward prefixes in the upper triangular part. The last row of the matrix equation states that perturbing the trace of a matrix perturbs the (negative of the) coefficient of z^{n-1} by the same amount.

If we further assume that the E is the backwards error matrix computed by a standard eigenvalue routine, then we might as well assume that E is nearly upper triangular. There will also be elements on the subdiagonal, and possibly on the next lower diagonal, depending on exactly which eigenvalue method is used.

A result analagous to Theorem 2.1 for matrix polynomials may be found in Section 4.

2.2 Geometry of Tangent Spaces and Transversality

Figure 1 illustrates matrix space ($R^{n \times n}$) with the usual Frobenius inner product:

$$(A, B) \equiv \mathrm{tr}(AB^T).$$

The basic players are

$p(z)$	a polynomial (not shown)
C	corresponding companion matrix
Orb	manifold of matrices similar to C
Tan	tangent space to this manifold $= \{CX - XC : X \in R^{n \times n}\}$
Normal	normal space to this manifold $= \{q(C^T) : q \text{ is a polynomial}\}$
Syl	"Sylvester" family of companion matrices

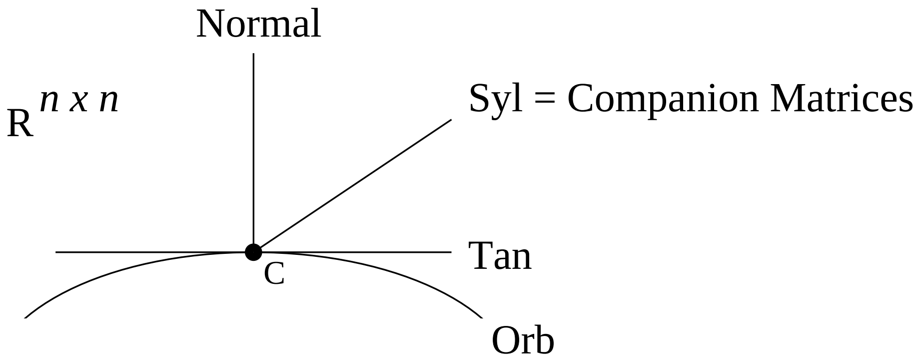

Figure 1 Matrix space near a companion matrix

- The curved surface represents the orbit of matrices similar to C. These are all the non-derogatory matrices that have eigenvalues equal to the roots of p with the same multiplicities. (The dimension of this space turns out to be $n^2 - n$. An informal argument is that only n parameters are specified, i.e., the eigenvalues.)
- The tangent space to this manifold, familiar to anyone who has studied elementary Lie algebra (or performed a first order calculation), is the set of *commutators* $\{CX - XC : X \in R^{n \times n}\}$. (Dimension $= n^2 - n$.)
- It is easy to check that if q is a polynomial, then any matrix of the form $q(C^T)$ is perpendicular to every matrix of the form $CX - XC$. Only slightly more difficult [1] is to verify that all matrices in the normal space are of the form $q(C^T)$. (Dimension $= n$.)
- The set of companion matrices (also known as the "Sylvester" family) is obviously an affine space through C. (Dimension $= n$.)

Proposition 2.1 *The Sylvester space of companion matrices is* transversal *to the tangent space i.e. every matrix may be expressed as a linear combination of a companion matrix and a matrix in the tangent space.*

This fact may be found in [1]. When we resolve E into components in these two directions, the former displays the change in the coefficients (to first order) and the latter does not affect the coefficients (to first order). Actually, a stronger statement holds: the resolution is unique. In the language of singularity theory, not only do we have transversality, but also *universality*: unique + transversal. We explicitly perform the resolution, and thereby prove the proposition, in the next subsection.

In numerical analysis, there are many examples where perturbations in "non-physical" directions cause numerical instability. In the companion matrix problem, only perturbations to the polynomial coefficients are relevant to the problem. Other perturbations are by-products of our numerical method. It is fortunate in this case that the error produced by an entire $n^2 - n$ dimensional space of extraneous directions is absorbed by the tangent space.

2.3 Algebra – The centralizer

Let us take a close look at the centralizer of C. This is the n dimensional linear space of polynomials in C, because C is non-derogatory. Taking the a_k as in (1), for $k = 0, \ldots, n-1$ and letting $a_n = 1$, $a_j = 0$ for $j \notin [0, n]$, we define matrices

$$M_k \equiv \sum_{j=k}^{n} a_j C^{j-k}.$$

Clearly $M_0 = 0, M_n = I$ and the M_k, $k = 1, \ldots n$ span the centralizer of C.

An interesting relationship (which is closely related to synthetic division) is that

$$p(t)(t - CI)^{-1} = \sum_{k=1}^{n} M_k t^{k-1}$$

[2, p. 85, Eq. (32)]. The trace of the above equation is the Newton identities in generating function form; a variation on the above equation gives the exact inverse of a Vandermonde matrix [10].

A more important relationship for our purposes is that

$$M_k = CM_{k+1} + a_k I, \quad k = 0, 1, 2, \ldots$$

from which we can inductively prove (the easiest way is backwards from $k = n$) that

$$\begin{array}{cc} & \begin{array}{cc} \qquad\qquad \underline{k} \qquad\qquad\qquad\qquad & \qquad \underline{n-k} \end{array} \\ M_k = & \left(\begin{array}{cccc|ccc} a_k & & & & -a_0 & & \\ a_{k+1} & a_k & & & -a_1 & \ddots & \\ \vdots & a_{k+1} & \ddots & & \vdots & \ddots & -a_0 \\ a_n = 1 & \vdots & \ddots & a_k & -a_{k-1} & \ddots & \vdots \\ & a_n = 1 & \ddots & a_{k+1} & & \ddots & \vdots \\ & & \ddots & \vdots & & & -a_{k-1} \\ & & & a_n = 1 & & & \end{array}\right). \end{array}$$

This is *almost* the Toeplitz matrix with (i, j) entry equal to $\pm a_{k+i-j}$ except that the left side of the matrix is lower triangular and the right side is strictly upper triangular.

We are now ready to resolve any E into

$$E = E^{\tan} + E^{\mathrm{syl}}. \tag{4}$$

All we need is the relationship that expresses how $E^{\tan}$ is perpendicular to the normal space:

$$\mathrm{tr}(M_k E^{\tan}) = 0 \text{ for } k = 1, \ldots, n.$$

We therefore conclude from (4) that

$$\mathrm{tr}(M_k E) = \mathrm{tr}(M_k E^{\mathrm{syl}}) = E^{\mathrm{syl}}_{k,n}. \tag{5}$$

Because of the almost Toeplitz nature of the M_k, the trace of $M_k E$ involves partial sums of elements of E along certain diagonals. Writing out this trace we have that

$$-E^{\mathrm{syl}}_{k,n} = \sum_{m=0}^{k-1} a_m \sum_{i=k+1}^{n} E_{i,i+m-k} - \sum_{m=k}^{n} a_m \sum_{i=1}^{k} E_{i,i+m-k}.$$

The above expression gives the coefficient of the perturbed characteristic polynomial correct to first order. (Since the coefficients are negated in (1), we are interested in $-E^{\mathrm{syl}}_{k,n}$.)

Since E^{syl} is zero in every column other than the last, we may also use (4) to calculate $E^{\tan}$, should we choose. $\square$

3 Numerical Experiments

3.1 A pair of 2x2 examples

The companion matrices

$$A = \begin{pmatrix} 0 & -1 \\ 1 & 2^k \end{pmatrix}, \quad \text{and } B = \begin{pmatrix} 0 & 1 \\ 1 & 2^{-3k} \end{pmatrix}$$

for k not too small illustrate many subtleties that occur in floating point arithmetic. For convenience, we assume our arithmetic follows the IEEE double precision standard for which $k = 27$ is large enough to illustrate our point.

To machine accuracy, the eigenvalues of A are 2^k and 2^{-k}. The eigenvalues of B are 1 and -1. LAPACK and MATLAB compute 2^k and 0 for the eigenvalues of A, while the eigenvalues of B are computed to be $1 - 2^{-52}$ and -1. Neither of these matrices are affected by balancing. Both of these answers are consistent with the error estimate in (2). Neither of these matrices gives answers with a small componentwise backward error. In the first case, the given product of the roots is 1 while the exact product of the computed roots is 0. In the second case, the given sum of the roots is 2^{-81}, while the exact sum of the computed roots is -2^{-52}.

However, MATLAB and LAPACK could be more accurate! Both packages compute the Schur form of a 2×2 matrix using an algorithm that is more unstable for the smaller eigenvalue than is necessary. We propose that such high quality packages should compute the eigenvalues of a general 2×2 matrix by solving the quadratic equation as accurately as possible given the rounded values of the trace and the determinant. If we have a 2×2 companion matrix, then there will be no roundoff error in the trace and the determinant.

The lesson of these examples is that the roots could be computed far more accurately than would be predicted by our error bound (2), but currently LAPACK and MATLAB return eigenvalues that are consistent with our bound. The other lesson is that without further assumptions it is impossible to require a small componentwise backward error. Fortunately, these examples are rather pathological. As we will see in the next subsection, in practice we do expect to compute roots with a small componentwise backwards error.

3.2 A more "realistic" set of tests

In this subsection we use Theorem 2.1 to predict the componentwise backward error. We also perform numerical experiments to measure the componentwise backward error. Our results show that Theorem 2.1 always predicts a small backward error and is most only pessimistic by one, two, or maybe three digits.

To predict the error, we must model the backwards error introduced by the QR algorithm. We decided to choose a model that is designed to compensate for pessimistic factors often found in rounding error analyses. Therefore, the model does not provide a rigorous bound, but at least in our test cases it seems to work well in practice.

What we do is consider an error matrix E with entries $\epsilon = 2^{-52}$ in all entries (i, j) with $j - i \geq -2$. For example, when $n = 6$,

$$E = \begin{pmatrix} \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ 0 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ 0 & 0 & \epsilon & \epsilon & \epsilon & \epsilon \\ 0 & 0 & 0 & \epsilon & \epsilon & \epsilon \end{pmatrix}.$$

This structure allows (with some overkill) for the possibility of double shifting in the eigenvalue algorithm. A dense perturbation matrix did not make a substantial difference in our test cases.

The standard algorithms balance the matrix by finding a diagonal matrix T such that $B = T^{-1}AT$ has a smaller norm than A. Our model will be that the eigenvalue algorithm computes the exact eigenvalues of a matrix $B + E'$, where $|E'| \leq E$, i.e. each element of E' has absolute value at most ϵ above the second lower diagonal and is 0 otherwise. Therefore we are computing the exact eigenvalues of $A + TE'T^{-1}$. To first order, then, the error in the coefficients is bounded by the absolute value of the matrix times the absolute value of the vector in the product (3), where the scans are computing using TET^{-1}. This is how we predict the δ_i. Further details appear in our MATLAB code in the Appendix.

Following [9] exactly, we explore the following degree 20 monic polynomials:

(1) "Wilkinson polynomial": zeros 1,2,3,...,20.

(2) the monic polynomial with zeros equally spaced in the interval $[-2.1, 1.9]$.

(3) $p(z) = (20!) \sum_{k=0}^{20} z^k/k!$.

(4) the Bernoulli polynomial of degree 20.

(5) the polynomial $z^{20} + z^{19} + z^{18} + \cdots + 1$.

(6) the monic polynomial with zeros $2^{-10}, 2^{-9}, 2^{-8}, \ldots, 2^9$.

(7) the Chebyshev polynomial of degree 20.

(8) the monic polynomial with zeros equally spaced on a sine curve, viz.,
$(2\pi/19(k + 0.5)) + i\sin(2\pi/19(k + 0.5)), k = -10, -9, -8, \ldots, 9$.

Our experimental results consist of three columns for each polynomial. To be precise, we first computed the coefficients either exactly or with 30 decimal precision using Mathematica. We then read these numbers into MATLAB which may not have rounded the last bit or two correctly.[1] Though we could have rounded the result correctly in Mathematica, we chose not to do so. Rather we took the rounded polynomials stored in MATLAB to be our "official" test cases.

Column 1: Log Predicted Error: In the first column we model the predicted error from (2) in the manner we described above. First we compute the modeled δa_i, and then display the rounded value of $\log_{10} |\delta a_i/a_i|$.

Column 2: Log Computed Error: We computed the eigenvalues using MATLAB, and then translated the IEEE double precision numbers to Mathematica without any error. We then computed the exact polynomial with these computed roots and compared the relative backward error in the coefficients.

Column 3: Pessimism Index: By taking the ratio of the computed error in column 2 to the predicted error described in column 1 and then taking the logarithm and rounding, we obtain a pessimism index. Indices such as 0,-1, and -2 indicate that we are pessimistic by at most two orders of magnitude; and index of -19 indicates a pessimism of 19 orders of magnitude. (Since we are using negative numbers, perhaps we should more properly call this an optimism index.)

There are no entries where the polynomial coefficient is zero. (The Bernoulli polynomial is a little funny in that it has a z^{19} term, but no other odd degree term.) The computed relative error would be infinite in many of these cases. The top coefficient is the log relative error in the determinant, i.e. the coefficient of the constant term; the bottom coefficient refers to the trace, i.e., the coefficient of t^{19}.

[1] Try entering `1.00000000000000018`, `1.00000000000000019`, and `1.000000000000000189999999` into MATLAB (fifteen zeros in each number). The results that we obtained on a SUN Sparc Station 10 were 1, $1 + 2^{-52}$, and $1 - 2^{-52}$ respectively, though the correctly rounded result should be $1 + 2^{-52}$ in all instances. Cleve Moler has responded that a better string parser is needed.

Relative errors in the Coefficients (log base 10)
for eight different degree 20 polynomials

Key to each panel in the table below		
Col 1	Col 2	Col 3
Predicted	Observed	Pessimism
Rel Error	Rel Error	Index

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
det→	-12 -13 -1	-12 -12 0	-13 -14 -1	-13 -14 -1	-14 -14 -1	-8 -14 -6	-12 -14 -2	-13 -14 -2	z^0
	-12 -14 -2	-13 -14 -1	-13 -14 -1		-13 -14 -1	-8 -14 -5			z^1
	-12 -13 -2	-12 -12 0	-13 -14 -1	-13 -15 -2	-13 -14 -1	-9 -14 -5	-12 -14 -2	-13 -15 -2	z^2
	-12 -13 -2	-13 -14 -1	-13 -14 -1		-13 -16 -3	-9 -14 -5			z^3
	-12 -15 -3	-12 -14 -2	-13 -14 -2	-13 -14 -1	-13 -14 -1	-9 -14 -5	-12 -14 -2	-13 -15 -2	z^4
	-12 -16 -4	-13 -14 -1	-13 -14 -1		-13 -14 -1	-10 -14 -4			z^5
	-12 -14 -2	-13 -14 -1	-13 -14 -1	-13 -14 -1	-13 -14 -1	-10 -15 -5	-13 -14 -1	-13 -15 -2	z^6
	-12 -15 -2	-13 -15 -2	-13 -14 -1		-13 -14 -1	-10 -14 -4			z^7
	-12 -15 -2	-13 -15 -2	-13 -15 -2	-13 -14 -1	-13 -14 -1	-11 -15 -5	-13 -14 -1	-13 -15 -1	z^8
	-13 -15 -3	-13 -15 -2	-13 -16 -3		-13 -14 -1	-11 -14 -3			z^9
	-13 -15 -2	-13 -14 -1	-13 -15 -2	-13 -14 -1	-13 -14 -1	-11 -14 -3	-13 -15 -1	-13 -15 -1	z^{10}
	-13 -15 -2	-13 -15 -1	-13 -15 -1		-13 -14 -1	-11 -14 -3			z^{11}
	-13 -15 -3	-13 -14 -1	-13 -15 -1	-13 -14 -1	-13 -14 -1	-12 -15 -4	-14 -15 -1	-14 -15 -1	z^{12}
	-13 -14 -1	-13 -14 -1	-13 -14 -1		-13 -14 -1	-12 -15 -3			z^{13}
	-13 -14 -1	-13 -14 -1	-13 -14 -1	-14 -14 -1	-13 -14 -1	-12 -15 -2	-14 -15 -1	-14 -14 -1	z^{14}
	-13 -15 -2	-13 -14 -1	-13 -14 -1		-13 -14 -1	-13 -14 -2			z^{15}
	-13 -15 -2	-14 -14 -1	-13 -14 -1	-13 -15 -2	-13 -15 -1	-13 -16 -3	-14 -15 -1	-14 -15 -1	z^{16}
	-13 -15 -2	-14 -14 -1	-13 -14 -1		-13 -15 -2	-13 -15 -2			z^{17}
	-14 -16 -2	-14 -15 -1	-14 -14 -1	-14 -16 -3	-13 -14 -1	-14 -15 -1	-14 -15 -1	-14 -15 -1	z^{18}
trace→	-14 -16 -2	-14 -14 0	-14 -14 -1	-14 -16 -2	-14 -14 0	-14 -16 -2			z^{19}

In all cases, we see that the computed backward relative error was excellent. With the exception of column (6), this is fairly well predicted usually with a pessimism index of one to three orders of magnitude. Column (6) is an exception, where the backward error is far more favorable than we predict.

Why this might be possible was explained in the previous subsection. We know the determinant to full precision (very rare when performing matrix computations!!). So long as our QR shifts and deflation criteria manage not to destroy the determinant, it will remain intact. In column (6) we see a case where this occurred. By contrast, in the previous subsection we illustrated a case where this did not occur.

We suspect that for any companion matrix, it is often if not always possible to choose shifts and deflation criteria to guarantee high (backward relative) accuracy even with the smallest of determinants, but we have not proven this.

4 Generalization to Matrix Polynomials

A monic matrix polynomial has the form

$$P(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1} + I_p x^n ,$$

where we assume that the coefficients A_i (and $A_n \equiv I_p$) are $p \times p$ matrices, and x is a scalar. It is of interest [3, 4, 5] to find the x for which $\det(P(x)) = 0$ and the corresponding eigenvectors $v(x)$ such that

$P(x)v(x) = 0$. Such information may be obtained from the $pn \times pn$ block companion matrix

$$C = \begin{pmatrix} 0 & & & & -A_0 \\ I_p & 0 & & & -A_1 \\ & I_p & 0 & & -A_2 \\ & & \ddots & & \vdots \\ & & & I_p & -A_{n-1} \end{pmatrix}. \tag{6}$$

The Sylvester space now has dimension np^2, while the tangent space to the orbit of C generically has dimension $n^2p^2 - np$, though it can be smaller. It seems that if $p > 1$, we have too many dimensions! We will now show that we may proceed in a manner that is analogous to that of Section 2.3 to obtain what is roughly the same answer, but to do so we must carefully pick a natural subspace of the tangent space that will give a universal decomposition. This is not necessary when $p = 1$. The natural subspace of the tangent space consists of all matrices of the form $CX - XC$ where the last p rows of X are 0.

Lemma 4.1 *Define*

$$M_k \equiv \sum_{j=k}^{n} C^{j-k}(I_n \otimes A_j),$$

where $\otimes$ denotes the Kronecker product and I_n is the identity matrix of order n. Then

$$M_k = CM_{k+1} + I_n \otimes A_k$$

and

$$M_k = \begin{array}{c} \begin{array}{cc} \underline{k} & \qquad\qquad\qquad \underline{n-k} \end{array} \\ \left(\begin{array}{cccc|ccc} A_k & & & & -A_0 & & \\ A_{k+1} & A_k & & & -A_1 & \ddots & \\ \vdots & A_{k+1} & \ddots & & \vdots & \ddots & -A_0 \\ A_n = I_p & \vdots & \ddots & A_k & -A_{k-1} & \ddots & \vdots \\ & A_n = I_p & \ddots & A_{k+1} & & \ddots & \vdots \\ & & \ddots & \vdots & & & -A_{k-1} \\ & & & A_n = I_p & & & \end{array} \right) \end{array}.$$

Proof These statements are readily verified by induction. □

We now introduce the $p \times p$ block trace of a matrx:

Definition 4.1 *If Z is a $pn \times pn$ matrix whose $p \times p$ blocks are denoted Z_{ij}, then we define*

$$tr_p(Z) \equiv \sum_{i=1}^{n} Z_{ii}.$$

Notice that $\mathrm{tr}_p(Z)$ is a $p \times p$ matrix, not a scalar.

Theorem 4.1 *Given the first $n-1$ block columns of a $pn \times pn$ matrix Z, the condition that*

$$0 = tr_p(ZM_k), \;\; k = 0, \ldots, n \tag{7}$$

is equivalent to the condition that

$$Z = CX - XC \text{ for some } X \text{ with } 0 \text{ bottom block row }. \tag{8}$$

Moreover, either condition determines the final block column of Z uniquely given the remaining columns.

Proof The (n,k) block entry of M_k is I_p and this determines Z_{kn} uniquely from (7). If X has 0 as its bottom block row, it is easy to verify that the map from X to the first $n-1$ block columns of $CX - XC$ has a trivial nullspace. Thus Z is uniquely determined by (8).

What remains is to show (8) implies (7). Suppose that Y has a zero bottom block row. Then $\mathrm{tr}_p(CY) = \mathrm{tr}_p(YC) = \sum_{i=1}^{n-1} Y_{i,i+1}$. Therefore, if X has a zero bottom block row, then $\mathrm{tr}_p(CXM_k) = \mathrm{tr}_p(XM_kC)$ by choosing $Y = XM_k$. Finally $\mathrm{tr}_p(XC^j(I_n \otimes A)C - XC^jC(I_n \otimes A)) = 0$ for any $p \times p$ matrix A, because $(I_n \otimes A)C - C(I_n \otimes A)$ is 0 except for the last block column. Therefore $XM_kC = XCM_k$, from which we conclude that $\mathrm{tr}_p(CXM_k) = \mathrm{tr}_p(XM_kC) = \mathrm{tr}_p(XCM_k)$. $\square$

We now summarize the geometry.

Corollary 4.1 *In the n^2p^2 dimensional space of $np \times np$ matrices, the $n^2p^2 - np^2$ subspace of the tangent space of the orbit of C defined either by (7) or (8) is transveral at C to the np^2 dimensional Sylvester space consisting of block companion matrices.*

We may now resolve any perturbation matrix E into

$$E = E^{\mathrm{tan}} + E^{\mathrm{syl}}. \tag{9}$$

as in (4), except now E^{tan} must be in this $n^2p^2 - np^2$ dimensional subspace, and E^{syl} is 0 except in the final *block* column. Because $\mathrm{tr}_p(ZM_k) \neq \mathrm{tr}_p(M_kZ)$, the correct result is that

$$-E^{\mathrm{syl}}_{k,n} = \sum_{m=0}^{k-1} \Big(\sum_{i=k+1}^{n} E_{i,i+m-k} \Big) A_m - \sum_{m=k}^{n} \Big(\sum_{i=1}^{k} E_{i,i+m-k} \Big) A_m.$$

The above expression gives the block coefficient of a perturbed matrix polynomial correct to first order.

$\square$

Acknowledgements

We would like to thank Jim Demmel for many interesting discussions and experiments on transversality and polynomial root finding.

References

[1] V.I. Arnold, On matrices depending on parameters, *Russian Math. Surveys 26* (1971), 29–43.

[2] F.R. Gantmacher, *The Theory of Matrices*, Vols. 1, 2, Chelsea, New York, 1959.

[3] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, 1982.

[4] D. Manocha and J. Demmel, Algorithms for intersecting parametric and algebraic curves I: simple intersections, *ACM Transactions on Graphics, 13*, (1994), 73–100.

[5] D. Manocha and J. Demmel, Algorithms for intersecting parametric and algebraic curves II: multiple intersections, *Computer graphics, vision and image processing: Graphical models and image processing*, to appear.

[6] C. Moler, Cleve's Corner: ROOTS – Of polynomials, that is, *The MathWorks Newsletter, v. 5 n. 1*, (Spring 1991), 8–9.

[7] B. Parlett and C. Reinsch, Balancing a matrix for calculation of eigenvalues and eigenvectors, *Numer. Math., 13*, (1969), 293–304.

[8] W.H. Press, et al., *Numerical Recipes*, Cambridge University Press, 1986.

[9] K. Toh and L.N. Trefethen, Pseudozeros of polynomials and pseudospectra of companion matrices, Cornell University Department of Computer Science preprint, TR 93-1360, June 1993 (to appear in Numerische Mathematik).

[10] J.F. Traub, Associated polynomials and uniform methods for the solution of linear problems, *SIAM Review, 8*, (1966), 277-301.

[11] P. Van Dooren and P. Dewilde, The eigenstructure of an arbitrary polynomial matrix: computational aspects, *Linear Alg. Appl. 50*, (1983), 545–579.

A MATLAB program used in experiments

The numerical part of our experiments was performed in MATLAB, while the exact component of the experiments was performed with Mathematica. We reproduce only the main MATLAB code below for the purpose of specifying precisely the crucial components of our experiments. The subroutines **scan** and **scanr** compute the plus-scan and the reversed plus-scan of a vector respectively.

```
% Predict and compute  the componentwise backward error in the eight
% polynomial test cases suggested by Toh and Trefethen.
%               --- Alan Edelman, October 1993

% Step 1 ... Run Mathematica Program to Compute Coefficients.
%            Output will be read into matlab as the array
%            d1  consisting of eight columns and
%            21 rows from the constant term (det) on top
%            to the x^19 term (trace), then 1 on the bottom
%            (Code not shown.)

% Step 2 ... Form the eight companion matrices
for i=1:8,
  eval(['m' num2str(i) '=zeros(20);']);
  eval(['m' num2str(i) '(2:21:380)=ones(1,19);']);
  eval(['m' num2str(i) '(:,20)=-d1(1:20,' num2str(i), ');']);
end

% Step 3 ... Obtain an "unbalanced" error matrix.
e=eps*ones(20);e=triu(e,-2);
for i=1:8,
  eval(['[t,b]=balance(m' num2str(i) ');']);
  ti=diag(1./diag(t));
  eval(['e' num2str(i) '=(t*e*ti)*norm(b);']);
end

% Step 4 ... Compute the first order perturbation matrix: er.
for j=1:8,
  eval(['e=e' num2str(j) ';']);
  forw=zeros(20);back=zeros(20);er=zeros(20,21);
  for i=1:19
    forw = forw + diag(scan(diag(e,i-1)),i-1);
    back = back + diag(scanr(diag(e,-i)),-i);
  end
  er(1:19,1:19)=back(2:20,1:19);er(:,2:21)=er(:,2:21)-forw;
  eval(['er' num2str(j) '=er;']);
end

% Step 5 ... Compute the predicted relative errors in the coefficients
predicted = zeros(20,8);
for i=1:8,
 eval(['predicted(:,' num2str(i) ')=abs(er' num2str(i) ')*abs(d1(:,i));']);
end
predicted=abs(predicted./d1(1:20,:));

% Step 6 ... Compute the exact relative errors in the coefficients
%            using Mathematica and finally display all relevant
%            quantities by taking the base 10 logarithm.  (Code not shown.)
```