MATHEMATICS OF COMPUTATION
Volume 85, Number 300, July 2016, Pages 1753–1774
http://dx.doi.org/10.1090/mcom/3020
Article electronically published on August 31, 2015

IMPLICIT QR FOR COMPANION-LIKE PENCILS

P. BOITO, Y. EIDELMAN, AND L. GEMIGNANI

Abstract. A fast implicit QR algorithm for eigenvalue computation of low rank corrections of unitary matrices is adjusted to work with matrix pencils arising from polynomial zero-finding problems. The modified QZ algorithm computes the generalized eigenvalues of certain $N \times N$ rank structured matrix pencils using $O(N^2)$ flops and $O(N)$ memory storage. Numerical experiments and comparisons confirm the effectiveness and the stability of the proposed method.

1. Introduction

Computing the roots of a univariate polynomial is a fundamental problem that arises in many applications. One way of numerically computing the roots of a polynomial is to form its companion matrix (pencil) and compute the (generalized) eigenvalues.

In a paper on polynomial root-finding [13], Jónsson and Vavasis present a comparative analysis of the accuracy of different matrix algorithms. The conclusion is that computing the roots of a polynomial by first forming the associated companion pencil $A - \lambda B$ and then solving $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ using the QZ algorithm provides better backward error bounds than computing the eigenvalues of the associated companion matrix by means of the QR algorithm. The analysis does not take into account the possible use of balancing which would have the effect of allineating the accuracy of the two approaches [2, 18]. However, it is worth noting that this use is not allowed if the customary QR and QZ algorithms are modified at the aim of reducing their complexity by one order of magnitude by exploiting the rank structure of the initial companion matrix (pencil). Thus, the use of the QZ algorithm applied to the companion pencil achieves the potential best score in terms of accuracy and efficiency among matrix methods for polynomial root-finding.

Another interesting appearance of the pencil approach is in the design of numerical routines for computing the determinant hence the zeros of polynomial matrices [17]. A classical technique is the interpolation of the determinant of the polynomial matrix $A(\lambda)$ at the roots of unity via FFT. This gives a representation of $p(\lambda) := \det A(\lambda)$ in the basis of Lagrange polynomials generated from the set of nodes. The main stumbling step in this process is the presence of undesirable infinite zeros or, equivalently, zero leading coefficients of $p(\lambda)$. This makes the companion matrix approach fully unusable. On the contrary, a companion pencil can still be formed whose generalized eigenvalues are the finite roots of the polynomial. If the pencil is $A - \lambda B$ directly constructed from the coefficients of the polynomial

Received by the editor January 29, 2014 and, in revised form, October 8, 2014 and December 6, 2014.

2010 *Mathematics Subject Classification.* Primary 65F15, 65H17.

This work was partially supported by MIUR, grant number 20083KLJEZ.

expressed in the Lagrange basis, then A is a unitary plus rank-one matrix and B is a rank-one perturbation of the identity matrix. By a preliminary reduction using the algorithm in [6] the matrices A and B can be transformed into upper Hessenberg and triangular forms, respectively.

This paper undertakes the task of developing an efficient variant of the QZ algorithm applied to a generalized companion pair (A, B), where A is upper Hessenberg, B is upper triangular and A and B are rank-one modifications of unitary matrices. These assumptions imply suitable rank structures in both A and B. We show that each matrix pair generated by the QZ process is also rank structured. By exploiting this property a novel, fast adaptation of the QZ eigenvalue algorithm is obtained which requires $O(n)$ flops per iteration and $O(n)$ memory space only. Numerical experiments confirm that the algorithm is stable.

The paper is organized as follows. In Section 2 we introduce the computational problem and describe the matrix structures involved. Section 3 and Section 4 deal with basic issues and concepts concerning the condensed representation and the invariance of these structures under the QZ process, whereas the main algorithm is presented in Section 5. Section 6 gives a backward error analysis for the QZ method applied to the companion pencil. In Section 7 we present an implementation of our fast variant of the QZ algorithm together with the results of extensive numerical experiments. Finally, the conclusion and a discussion are the subjects of Section 8.

2. The problem statement

Companion pencils and generalized companion pencils expressed in the Lagrange basis at the roots of unity are specific instances of the following general class.

Definition 2.1. The matrix pair (A, B), $A, B \in \mathbb{C}^{N\times N}$, belongs to the class $\mathcal{P}_N \subset \mathcal{C}^{N\times N} \times \mathcal{C}^{N\times N}$ of generalized companion pencils iff:

(1) $A \in \mathbb{C}^{N\times N}$ is upper Hessenberg.
(2) $B \in \mathbb{C}^{N\times N}$ is upper triangular.
(3) There exist two vectors $\boldsymbol{z} \in \mathbb{C}^N$ and $\boldsymbol{w} \in \mathbb{C}^N$ and a unitary matrix $V \in \mathbb{C}^{N\times N}$ such that

$$A = V - \boldsymbol{z}\boldsymbol{w}^*. \tag{2.1}$$

(4) There exist two vectors $\boldsymbol{p} \in \mathbb{C}^N$ and $\boldsymbol{q} \in \mathbb{C}^N$ and a unitary matrix $U \in \mathbb{C}^{N\times N}$ such that

$$B = U - \boldsymbol{p}\boldsymbol{q}^*. \tag{2.2}$$

In order to characterize the individual properties of the matrices A and B we give some additional definitions.

Definition 2.2. We denote by $\mathcal{T}_N$ the class of upper triangular matrices $B \in \mathbb{C}^{N\times N}$ which are rank-one perturbations of unitary matrices, i.e., such that (2.2) holds for a suitable unitary matrix U and vectors $\boldsymbol{p}, \boldsymbol{q}$.

Since B is upper triangular the strictly lower triangular part of the unitary matrix U in (2.2) coincides with the corresponding part of the rank-one matrix $\boldsymbol{p}\boldsymbol{q}^*$, i.e.,

$$U(i, j) = p(i)q^*(j), \quad 1 \le j < i \le N, \tag{2.3}$$

where $\{p(i)\}_{i=1,\dots,N}$ and $\{q(j)\}_{j=1,\dots,N}$ are the entries of $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively.

Definition 2.3. We denote by $\mathcal{U}_N$ the class of unitary matrices $U \in \mathbb{C}^{N\times N}$ which satisfy the condition (2.3), i.e., for which there exist vectors $\boldsymbol{p}, \boldsymbol{q}$ such that the matrix $B = U - \boldsymbol{p}\boldsymbol{q}^*$ is an upper triangular matrix.

Observe that we have

$$U \in \mathcal{U}_N \Rightarrow \operatorname{rank} U(k+1\colon N, 1\colon k) \le 1, \quad k = 1, \dots, N-1.$$

From the nullity theorem [11] (see also [9, p. 142]) it follows that the same property also holds in the strictly upper triangular part, namely,

$$U \in \mathcal{U}_N \Rightarrow \operatorname{rank} U(1\colon k, k+1\colon N) \le 1, \quad k = 1, \dots, N-1. \tag{2.4}$$

Definition 2.4. We denote by $\mathcal{H}_N$ the class of upper Hessenberg matrices $A \in \mathbb{C}^{N\times N}$ which are rank-one perturbations of unitary matrices, i.e., such that (2.1) holds for a suitable unitary matrix V and vectors $\boldsymbol{z}, \boldsymbol{w}$.

Definition 2.5. We denote by $\mathcal{V}_N$ the class of unitary matrices $V \in \mathbb{C}^{N\times N}$ for which there exist vectors $\boldsymbol{z}, \boldsymbol{w}$ such that the matrix $A = V - \boldsymbol{z}\boldsymbol{w}^*$ is an upper Hessenberg matrix.

We find that

$$V \in \mathcal{V}_N \Rightarrow \operatorname{rank} V(k+2\colon N, 1\colon k) \le 1, \quad k = 1, \dots, N-2.$$

Again, from the nullity theorem, it follows that a similar property also holds in the upper triangular part, namely,

$$V \in \mathcal{V}_N \Rightarrow \operatorname{rank} V(1\colon k, k\colon N) \le 2, \quad k = 1, \dots, N. \tag{2.5}$$

The QZ algorithm is the customary method for solving generalized eigenvalue problems numerically by means of unitary transformations (see e.g. [12] and [20]). Recall that the Hessenberg and triangular forms are preserved under the QZ iteration; an easy computation then yields

$$(A, B) \in \mathcal{P}_N,\ (A, B) \overset{\text{QZ step}}{\to} (A_1, B_1) \Rightarrow (A_1, B_1) \in \mathcal{P}_N. \tag{2.6}$$

Indeed if Q and Z are unitary then from (2.1) and (2.2) it follows that the matrices $A_1 = Q^*AZ$ and $B_1 = Q^*BZ$ satisfy the relations

$$A_1 = V_1 - \boldsymbol{z}_1\boldsymbol{w}_1^*, \quad B_1 = U_1 - \boldsymbol{p}_1\boldsymbol{q}_1^*$$

with the unitary matrices $V_1 = Q^*VZ$, $U_1 = Q^*UZ$ and the vectors $\boldsymbol{z}_1 = Q^*\boldsymbol{z}$, $\boldsymbol{w}_1 = Z^*\boldsymbol{w}$, $\boldsymbol{p}_1 = Q^*\boldsymbol{p}$, $\boldsymbol{q}_1 = Z^*\boldsymbol{q}$. Moreover, one can choose the unitary matrices Q and Z such that the matrix A_1 is upper Hessenberg and the matrix B_1 is upper triangular. Thus, one can in principle think of designing a structured QZ iteration that, given in input a condensed representation of the matrix pencil $(A, B) \in \mathcal{P}_N$, returns as output a condensed representation of $(A_1, B_1) \in \mathcal{P}_N$ generated by one step of the classical QZ algorithm applied to (A, B). In the next sections we first introduce an eligible representation of a rank-structured matrix pencil $(A, B) \in \mathcal{P}_N$ and then discuss the modification of this representation under the QZ process.

3. Quasiseparable representations

In this section we present the properties of quasiseparable representations of rank-structured matrices [8], [9, Chapters 4,5]. First we recall some general results and definitions. Subsequently, we describe their adaptations for the representation of the matrices involved in the structured QZ iteration applied to an input matrix pencil $(A, B) \in \mathcal{P}_N$.

3.1. **Preliminaries.** A matrix $M = \{M_{ij}\}_{i,j=1}^{N}$ is (r^L, r^U)-*quasiseparable*, with r^L, r^U positive integers if, using MATLAB[1] notation,

$$\max_{1\le k\le N-1} \operatorname{rank}(M(k+1:N, 1:k)) \le r^L,$$
$$\max_{1\le k\le N-1} \operatorname{rank}(M(1:k, k+1:N)) \le r^U.$$

Roughly speaking, this means that every submatrix extracted from the lower triangular part of M has rank at most r^L, and every submatrix extracted from the upper triangular part of M has rank at most r^U. Under this hypothesis, M can be represented using $\mathcal{O}(((r^L)^2 + (r^U)^2)N)$ parameters. In this subsection we present such a representation.

The quasiseparable representation of a rank-structured matrix consists of a set of vectors and matrices used to generate its entries. For the sake of notational simplicity, generating matrices and vectors are denoted by a lower-case letter.

In this representation, the entries of M take the form

$$M_{ij} = \begin{cases} p(i)a_{ij}^{>}q(j), & 1 \le j < i \le N, \\ d(i), & 1 \le i = j \le N, \\ g(i)b_{ij}^{<}h(j), & 1 \le i < j \le N, \end{cases} \tag{3.1}$$

where:

- $p(2), \dots, p(N)$ are row vectors of length r^L, $q(1), \dots, q(N-1)$ are column vectors of length r^L, and $a(2), \dots, a(N-1)$ are matrices of size $r^L \times r^L$; these are called *lower quasiseparable generators* of order r^L;
- $d(1), \dots, d(N)$ are numbers (the diagonal entries),
- $g(2), \dots, g(N)$ are row vectors of length r^U, $h(1), \dots, h(N-1)$ are column vectors of length r^U, and $b(2), \dots, b(N-1)$ are matrices of size $r^U \times r^U$; these are called *upper quasiseparable generators* of order r^U;
- the matrices $a_{ij}^{>}$ and $b_{ij}^{<}$ are defined as

$$\begin{cases} a_{ij}^{>} = a(i-1)\cdots a(j+1) \text{ for } i > j+1, \\ a_{j+1,j}^{>} = 1 \end{cases}$$

and

$$\begin{cases} b_{ij}^{<} = b(i+1)\cdots b(j-1) \text{ for } j > i+1, \\ b_{i,i+1}^{<} = 1. \end{cases}$$

From (2.4) it follows that any matrix from the class $\mathcal{U}_N$ has upper quasiseparable generators with orders equal to one.

The quasiseparable representation can be generalized to the case where M is a block matrix, and to the case where the generators do not all have the same size, provided that their product is well defined. Each block M_{ij} of size $m_i \times n_j$ is represented as in (3.1), except that the sizes of the generators now depend on m_i and n_j, and possibly on the index of a and b. More precisely:

- $p(i), q(j), a(k)$ are matrices of sizes $m_i \times r_{i-1}^L$, $r_j^L \times n_j$, $r_k^L \times r_{k-1}^L$, respectively;
- $d(i)$ $(i = 1, \dots, N)$ are $m_i \times n_i$ matrices;
- $g(i), h(j), b(k)$ are matrices of sizes $m_i \times r_i^U$, $r_{j-1}^U \times n_j$, $r_{k-1}^U \times r_k^U$, respectively.

[1]MATLAB is a registered trademark of The Mathworks, Inc.

The numbers r_k^L, r_k^U $(k = 1, \dots, N-1)$ are called the *orders* of these generators.

It is worth noting that lower and upper quasiseparable generators of a matrix are not uniquely defined. A set of generators with minimal orders can be determined according to the ranks of maximal submatrices located in the lower and upper triangular parts of the matrix.

One advantage of the block representation for the purposes of the present paper consists in the fact that $N \times N$ upper Hessenberg matrices can be treated as $(N + 1 \times N + 1)$ block upper triangular matrices by choosing blocks of the following sizes:

$$m_1 = \cdots = m_N = 1,\ m_{N+1} = 0, \quad n_1 = 0,\ n_2 = \cdots = n_{N+1} = 1. \tag{3.2}$$

Such a treatment also allows us to consider quasiseparable representations which include the main diagonals of matrices. Assume that C is an $N \times N$ scalar matrix with the entries in the upper triangular part represented in the form

$$C(i,j) = g(i) b_{i-1,j}^{<} h(j), \quad 1 \le i \le j \le N \tag{3.3}$$

with matrices $g(i), h(i)$ $(i = 1, \dots, N)$, $b(k)$ $(k = 1, \dots, N-1)$ of sizes $1 \times r_i, r_i \times 1, r_k \times r_{k+1}$. The elements $g(i), h(i)$ $(i = 1, \dots, N)$, $b(k)$ $(k = 1, \dots, N-1)$ are called *upper triangular generators* of the matrix C with orders r_k $(k = 1, \dots, N)$. From (2.5) it follows that any matrix from the class $\mathcal{V}_N$ has upper triangular generators with orders not greater than two. If we treat a matrix C as a block one with entries of sizes (3.2) we conclude that the elements $g(i)$ $(i = 1, \dots, N)$, $h(j-1)$ $(j = 2, \dots, N+1)$, $b(k-1)$ $(k = 2, \dots, N)$ are upper quasiseparable generators of C.

Matrix operations involving zero-dimensional arrays (empty matrices) are defined according to the rules used in MATLAB and described in [3]. In particular, the product of an $m \times 0$ matrix by a $0 \times m$ matrix is an $m \times m$ matrix with all entries equal to 0. Empty matrices may be used in assignment statements as a convenient way to add and/or delete rows or columns of matrices.

3.2. **Representations of matrix pairs from the class $\mathcal{P}_N$.** Let (A, B) be a matrix pair from the class $\mathcal{P}_N$. The corresponding matrix A from the class $\mathcal{H}_N$ is completely defined by the following parameters:

(1) the subdiagonal entries σ_k^A $(k = 1, \dots, N-1)$ of the matrix A;
(2) the upper triangular generators $g_V(i), h_V(i)$ $(i = 1, \dots, N)$, $b_V(k)$ $(k = 1, \dots, N-1)$ of the corresponding unitary matrix V from the class $\mathcal{V}_N$;
(3) the vectors of perturbation $\boldsymbol{z} = \operatorname{col}(z(i))_{i=1}^N$, $\boldsymbol{w} = \operatorname{col}(w(i))_{i=1}^N$.

From (2.5) it follows that the matrix $V \in \mathcal{V}_N$ has upper triangular generators with orders not greater than two.

The corresponding matrix B from the class $\mathcal{T}_N$ is completely defined by the following parameters:

(1) the diagonal entries $d_B(k)$ $(k = 1, \dots, N)$ of the matrix B;
(2) the upper quasiseparable generators $g_U(i)$ $(i = 1, \dots, N-1)$, $h_U(j)$ $(j = 2, \dots, N)$, $b_U(k)$ $(k = 2, \dots, N-1)$ of the corresponding unitary matrix U from the class $\mathcal{U}_N$;
(3) the vectors of perturbation $\boldsymbol{p} = \operatorname{col}(p(i))_{i=1}^N$, $\boldsymbol{q} = \operatorname{col}(q(i))_{i=1}^N$.

From (2.4) it follows that the matrix $U \in \mathcal{U}_N$ has upper quasiseparable generators with orders equal to one.

All the given parameters define completely the matrix pair (A, B) from the class $\mathcal{P}_N$.

Each step of structured QZ should update these parameters while maintaining the minimal orders of generators. However, structured algorithms for the multiplication of quasiseparable matrices may output redundant generators for the product matrix. For this reason, we will need an algorithm that compresses generators to minimal order. The algorithm we use is derived from previous work and is described in detail in the Appendix.

4. The QZ step via generators

4.1. **Classical QZ.** Let (A, B) be a pair of $N \times N$ matrices, with $A = (a_{ij})_{i,j=1}^{N}$ upper Hessenberg and $B = (b_{ij})_{i,j=1}^{N}$ upper triangular. The implicit QZ step applied to (A, B) consists in the computation of unitary matrices Q and Z such that

$$\begin{array}{c} \text{the matrix } A_1 = Q^* A Z \text{ is upper Hessenberg;} \\ \text{the matrix } B_1 = Q^* B Z \text{ is upper triangular;} \end{array} \tag{4.1}$$

and, in addition, some initial conditions are satisfied. In the case of the single-shift implicit QZ step, the unitary matrices Q and Z are upper Hessenberg and take the form

$$Q = \tilde{Q}_1 \tilde{Q}_2 \cdots \tilde{Q}_{N-1}, \quad Z = \tilde{Z}_1 \tilde{Z}_2 \cdots \tilde{Z}_{N-1}, \tag{4.2}$$

where

$$\tilde{Q}_i = I_{i-1} \oplus Q_i \oplus I_{N-i-1}, \quad \tilde{Z}_i = I_{i-1} \oplus Z_i \oplus I_{N-i-1} \tag{4.3}$$

and Q_i, Z_i are complex Givens rotation matrices. The first Givens matrix Q_1 is chosen so that

$$Q_1^* \begin{pmatrix} a_{11} - \alpha \\ a_{21} \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}, \tag{4.4}$$

where the shift $\alpha \in \mathbb{C}$ is an approximation of the eigenvalue that is currently being computed. The next Givens matrices Q_i and Z_i are computed and applied to the pencil using a bulge-chasing technique, which we show graphically on a 4×4 example (see e.g., [12]):

$$\begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix} \xrightarrow{Q_1^* \cdot} \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix}$$

$$\xrightarrow{\cdot Z_1} \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix} \xrightarrow{Q_2^* \cdot} \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix}$$

$$\xrightarrow{\cdot Z_2} \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix} \xrightarrow{Q_3^* \cdot} \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}$$

$$\xrightarrow{\cdot Z_3} \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}, \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix}.$$

We provide for further reference a pseudocode description of a bulge-chasing sweep:

(1) determine Q_1^* from (4.4) and compute $A \leftarrow Q_1^* A$
(2) **for** $k = 1 : N-2$ **do**
(3) $\quad B \leftarrow Q_k^* B$
(4) $\quad$ determine Z_k that eliminates bulge in B
(5) $\quad A \leftarrow A * Z_k$
(6) $\quad B \leftarrow B * Z_k$
(7) $\quad$ determine Q_{k+1}^* that eliminates bulge in A
(8) $\quad A \leftarrow Q_{k+1}^* A$
(9) **end do**
(10) $B \leftarrow Q_{N-1}^* B$
(11) determine Z_{N-1} that eliminates bulge in B
(12) $A \leftarrow A * Z_{N-1}$
(13) $B \leftarrow B * Z_{N-1}$.

4.2. **Structured QZ.** We present now a fast adaptation of the implicit single-shift QZ algorithm for an input matrix pair $(A, B) \in \mathcal{P}_N$. The modified algorithm works on the generators of the two matrices and this explains why it is referred to as a structured implicit QZ iteration with single shift. More specifically, the input pair $(A, B) = (A_0, B_0)$ is first represented by means of a linear set of generators as explained in Subsection 3.2. Then a high-level description of structured QZ iteration goes as follows.

(1) Given the number $\alpha \in \mathbb{C}$, perform one step of the implicit single-shift QZ algorithm (2.6) by computing subdiagonal entries of the matrix A_1, diagonal entries of the matrix B_1, vectors of perturbation $\boldsymbol{z}_1, \boldsymbol{w}_1, \boldsymbol{p}_1, \boldsymbol{q}_1$ as well as upper triangular generators of the matrix V_1 and upper quasiseparable generators of the matrix U_1 (with redundant orders).
(2) Compress the representations of V_1 and U_1 by using the compression algorithm in the Appendix.

Let us see how the QZ computation at step 1 above is efficiently performed by working on the set of generators. When appropriate, we will reference lines from the classical QZ pseudocode shown above.

For the most part, the structured update process is carried out separately on V, U, $\boldsymbol{z}$ and $\boldsymbol{w}$. However, note the following correspondences between some quantities computed in the structured algorithm below and some quantities used in the classical approach:

- The matrices Φ_k at step 3a below correspond to the bulges created in the matrix B and are explicitly computed in order to determine the Givens matrices Z_k.
- Analogously, the matrices Ω_k at step 3b below correspond to the bulges created in the matrix A and are explicitly computed in order to determine the Givens matrices Q_{k+1}.
- The diagonal entries of B and the subdiagonal entries of A are explicitly stored in d_B and σ_A, respectively. The diagonal entries of A, on the other hand, can be easily computed from the generators. For instance, the quantity $\epsilon_1 = A(1,1)$ is computed at step 1 in order to determine the first Givens

matrix Q_1 that encodes the shift. Observe that the explicit computation of diagonal and subdiagonal entries is also crucial when performing deflation.

Structured QZ bulge-chasing algorithm.

Input: the shift $\alpha \in \mathbb{C}$, subdiagonal entries σ_k^A for the matrix A, upper triangular generators $g_V(k)$, $h_V(k)$, $b_V(k)$ for the matrix V, diagonal entries $d_B(k)$ for the matrix B, upper quasiseparable generators $g_U(k)$, $h_U(k)$, $b_U(k)$ for the matrix U, and perturbation vectors $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{z}$ and $\boldsymbol{w}$.

Output: subdiagonal entries $\sigma_k^{A_1}$ for the matrix A_1, upper triangular generators $g_{V_1}(k)$, $h_{V_1}(k)$, $b_{V_1}(k)$ of redundant orders for the matrix V_1, diagonal entries $d_{B_1}(k)$ for the matrix B_1, upper quasiseparable generators $g_{U_1}(k)$, $h_{U_1}(k)$, $b_{U_1}(k)$ of redundant orders for the matrix U_1, perturbation vectors $\boldsymbol{p}_1$, $\boldsymbol{q}_1$, $\boldsymbol{z}_1$, $\boldsymbol{w}_1$, and, if needed, the matrices Q_k and Z_k.

(1) Compute

$$\sigma_1^V = \sigma_1^A + z(2)w^*(1), \quad \epsilon_1 = g_V(1)h_V(1) - z(1)w^*(1).$$

(Pseudocode line 1). Determine a complex Givens transformation matrix Q_1 from the condition

$$Q_1^* \begin{pmatrix} \epsilon_1 - \alpha \\ \sigma_1^A \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}.$$

Compute

$$\Gamma_2 = Q_1^* \begin{pmatrix} g_V(1)h_V(1) & g_V(1)b_V(1) \\ \sigma_1^V & g_V(2) \end{pmatrix}$$

and determine the matrices $\tilde{g}_V(2), \beta_2^V, f_2^V, \phi_2^V$ of sizes $1\times(r_2^V+1), 1\times(r_2^V+1), 1\times 1, 1\times r_2^V$ from the partitions

$$\Gamma_2 = \begin{bmatrix} \tilde{g}_V(2) \\ \beta_2^V \end{bmatrix}, \quad \beta_2^V = \begin{pmatrix} f_2^V & \phi_2^V \end{pmatrix}.$$

Compute

$$\begin{pmatrix} z^{(1)}(1) \\ \chi_2 \end{pmatrix} = Q_1^* \begin{pmatrix} z(1) \\ z(2) \end{pmatrix}$$

with the numbers $z^{(1)}(1), \chi_2$.

Compute

$$f_2^A = f_2^V - \chi_2 w^*(1), \quad \varphi_2^A = \phi_2^V.$$

(2) Set

$$\gamma_1 = w(1),\ c_1 = p(1),\ \theta_1 = q(1).$$

Compute

$$d_U(1) = d_B(1) + p(1)q^*(1)$$

and set

$$f_1^U = d_U(1),\ \phi_1^U = g_U(1),$$

$$f_1^B = d_B(1), \quad \varphi_1 = g_U(1).$$

(3) For $k = 1, \ldots, N-2$ perform the following.

(a) *(Pseudocode line 3).* Compute the numbers

$$\varepsilon_k = \varphi_k h_U(k+1) - c_k q^*(k+1),$$
$$\epsilon_{k+1} = \varphi_k^A h_V(k+1) - \chi_k w^*(k+1),$$
$$d_U(k+1) = d_B(k+1) + p(k+1)q^*(k+1),$$

and the 2×2 matrix Φ_k by the formula

$$\Phi_k = Q_k^* \begin{pmatrix} f_k^B & \varepsilon_k \\ 0 & d_B(k+1) \end{pmatrix}.$$

(b) *(Pseudocode line 4).* Determine a complex Givens rotation matrix Z_k such that

$$\Phi_k(2,:)Z_k = \begin{pmatrix} 0 & * \end{pmatrix}.$$

Compute the 2×2 matrix Ω_k by the formula

$$\Omega_k = \begin{pmatrix} f_{k+1}^A & \epsilon_{k+1} \\ 0 & \sigma_{k+1}^A \end{pmatrix} Z_k.$$

(c) *(Pseudocode line 7).* Determine a complex Givens rotation matrix Q_{k+1} and the number $\sigma_k^{A_1}$ such that

$$Q_{k+1}^* \Omega_k(:,1) = \begin{pmatrix} \sigma_k^{A_1} \\ 0 \end{pmatrix}.$$

(d) *(Pseudocode lines 5 and 8).* Compute

$$\Gamma'_{k+2} = Q_{k+1}^* \begin{pmatrix} f_{k+1}^V & \phi_{k+1}^V h_V(k+1) & \phi_{k+1}^V b_V(k+1) \\ z(k+2)\gamma_k^* & \sigma_{k+1}^V & g_V(k+2) \end{pmatrix},$$

$$\Gamma_{k+2} = \Gamma'_{k+2} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+2}^V} \end{pmatrix},$$

$$C_{k+2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & h_V(k+1) & b_V(k+1) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+2}^V} \end{pmatrix}$$

and determine the matrices $\tilde{d}_{V_1}(k+2), \tilde{g}_{V_1}(k+2), \beta_{k+2}^V, f_{k+2}^V, \phi_{k+2}^V$ of sizes $1\times 1, 1\times(r_{k+2}^V+1), 1\times(r_{k+2}^V+1), 1\times 1, 1\times r_{k+2}^V$ from the partitions

$$\Gamma_{k+2} = \begin{bmatrix} \tilde{d}_{V_1}(k+2) & \tilde{g}_{V_1}(k+2) \\ * & \beta_{k+2}^V \end{bmatrix}, \quad \beta_{k+2}^V = \begin{bmatrix} f_{k+2}^V & \phi_{k+2}^V \end{bmatrix}$$

and the matrices $\tilde{h}_{V_1}(k+2), \tilde{b}_{V_1}(k+2)$ of sizes $(r_{k+1}^V+1)\times 1, (r_{k+1}^V+1)\times(r_{k+2}^V+1)$ from the partition

$$C_{k+2} = \begin{pmatrix} \tilde{h}_{V_1}(k+2) & \tilde{b}_{V_1}(k+2) \end{pmatrix}.$$

(e) *(Pseudocode lines 3 and 6).* Compute

$$\Lambda_{k+1} =$$
$$Q_k^* \begin{pmatrix} f_k^U & \phi_k^U h_U(k+1) & \phi_k^U b_U(k+1) \\ p(k+1)\theta_k^* & d_U(k+1) & g_U(k+1) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+1}^U} \end{pmatrix},$$

$$D_{k+1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & h_U(k+1) & b_U(k+1) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+1}^U} \end{pmatrix},$$

and determine the matrices $\tilde{d}_{U_1}(k), \tilde{g}_{U_1}(k), \beta^U_{k+1}, f^U_{k+1}, \phi^U_{k+1}$ of sizes $1\times 1, 1\times(r^U_{k+1}+1), 1\times(r^U_{k+1}+1), 1\times 1, 1\times r^U_{k+1}$ from the partitions

$$\Lambda_{k+1} = \begin{pmatrix} \tilde{d}_{U_1}(k) & \tilde{g}_{U_1}(k) \\ * & \beta^U_{k+1} \end{pmatrix}, \quad \beta^U_{k+1} = \begin{pmatrix} f^U_{k+1} & \phi^U_{k+1} \end{pmatrix}$$

and the matrices $\tilde{h}_{U_1}(k+1), \tilde{b}_{U_1}(k+1)$ of sizes $(r^U_k+1)\times 1, (r^U_k+1)\times(r^U_{k+1}+1)$ from the partition

$$D_{k+1} = \begin{pmatrix} \tilde{h}_{U_1}(k+1) & \tilde{b}_{U_1}(k+1) \end{pmatrix}.$$

(f) *(Update perturbation vectors for A).* Compute

$$\begin{pmatrix} z^{(1)}(k+1) \\ \chi_{k+2} \end{pmatrix} = Q^*_{k+1} \begin{pmatrix} \chi_k \\ z(k+1) \end{pmatrix},$$
$$\begin{pmatrix} w^{(1)}(k) \\ \gamma_{k+1} \end{pmatrix} = Z^*_k \begin{pmatrix} \gamma_k \\ w(k+1) \end{pmatrix}$$

with the numbers $z^{(1)}(k+1), w^{(1)}(k), \chi_{k+2}, \gamma_{k+1}$.

(g) *(Update perturbation vectors for B).* Compute

$$\begin{pmatrix} q^{(1)}(k) \\ \theta_{k+1} \end{pmatrix} = Z^*_k \begin{pmatrix} \theta_k \\ q(k+1) \end{pmatrix}, \quad \begin{pmatrix} p^{(1)}(k) \\ c_{k+1} \end{pmatrix} = Q^*_k \begin{pmatrix} c_k \\ p(k+1) \end{pmatrix}$$

with the numbers $q^{(1)}(k), p^{(1)}(k), \theta_{k+1}, c_{k+1}$.

(h) Compute

$$f^A_{k+2} = f^V_{k+2} - \chi_{k+2}\gamma^*_{k+1}, \quad \varphi^A_{k+2} = \phi^V_{k+2}.$$

Compute

$$f^B_{k+1} = f^U_{k+1} - c_{k+1}\theta^*_{k+1}, \quad \varphi_{k+1} = \phi^U_{k+1}.$$

(4) *(Pseudocode line 10).* Compute the numbers

$$\varepsilon_{N-1} = \varphi_{N-1}h_U(N) - c_{N-1}q^*(N), \quad d_U(N) = d_B(N) + p(N)q^*(N)$$

and the 2×2 matrix Φ_{N-1} by the formula

$$\Phi_{N-1} = Q^*_{N-1} \begin{pmatrix} f^B_{N-1} & \varepsilon_{N-1} \\ 0 & d_B(N) \end{pmatrix}.$$

(5) *(Pseudocode line 11).* Determine a complex Givens rotation matrix Z_{N-1} such that

$$\Phi_{N-1}(2,:)Z_{N-1} = \begin{pmatrix} 0 & * \end{pmatrix}.$$

(6) *(Pseudocode line 12).* Compute

$$\Gamma_{N+1} = \begin{pmatrix} f^V_N & \phi^V_N h_V(N) \end{pmatrix} Z_{N-1},$$
$$C_{N+1} = \begin{pmatrix} 1 & 0 \\ 0 & h_V(N) \end{pmatrix} Z_{N-1}, \quad \tilde{h}^{(1)}_V(N) = 1.$$

and determine the numbers $\tilde{d}_{V_1}(N+1), \tilde{g}_{V_1}(N+1)$ from the partition

$$\Gamma_{N+1} = \begin{bmatrix} \tilde{d}_{V_1}(N+1) & \tilde{g}_{V_1}(N+1) \end{bmatrix}$$

and r^V_N+1-dimensional columns $\tilde{h}_{V_1}(N+1), \tilde{b}_{V_1}(N+1)$ from the partition

$$C_{N+1} = \begin{pmatrix} \tilde{h}_{V_1}(N+1) & \tilde{b}_{V_1}(N+1) \end{pmatrix}.$$

(7) *(Pseudocode line 13).* Compute

$$\Lambda_N = Q^*_{N-1}\begin{pmatrix} f^U_{N-1} & \phi^U_{N-1}h_U(N) \\ p(N)\theta^*_{N-1} & d_U(N) \end{pmatrix} Z_{N-1},$$

$$D_N = \begin{pmatrix} 1 & 0 \\ 0 & h_U(N) \end{pmatrix} Z_{N-1}, \quad \tilde{h}^{(1)}_U(N) = 1$$

and determine the numbers $\tilde{d}_{U_1}(N-1), \tilde{g}_{U_1}(N-1), \tilde{d}_{U_1}(N)$ from the partition

$$\Lambda_N = \begin{bmatrix} \tilde{d}_{U_1}(N-1) & \tilde{g}_{U_1}(N-1) \\ * & \tilde{d}_{U_1}(N) \end{bmatrix}$$

and $r^U_{N-1}+1$-dimensional columns $\tilde{h}_{U_1}(N), \tilde{b}_{U_1}(N)$ from the partition

$$D_N = \begin{pmatrix} \tilde{h}_{U_1}(N) & \tilde{b}_{U_1}(N) \end{pmatrix}.$$

Set

$$\tilde{h}_{V_1}(N+1) = 1, \quad \tilde{h}_{U_1}(N+1) = 1.$$

(8) *(Update perturbation vectors).* Compute

$$\begin{pmatrix} w^{(1)}(N-1) \\ w^{(1)}(N) \end{pmatrix} = Q^*_{N-1}\begin{pmatrix} \gamma_{N-1} \\ w(N) \end{pmatrix},$$

$$\begin{pmatrix} p^{(1)}(N-1) \\ p^{(1)}(N) \end{pmatrix} = Q^*_{N-1}\begin{pmatrix} c_{N-1} \\ p(N) \end{pmatrix}, \begin{pmatrix} q^{(1)}(N-1) \\ q^{(1)}(N) \end{pmatrix} = Z^*_{N-1}\begin{pmatrix} \theta_{N-1} \\ q(N) \end{pmatrix}.$$

Set

$$z^{(1)}(N) = \chi_N$$

and compute

$$\sigma^{A_1}_{N-1} = \tilde{d}_{V_1}(N+1) - z^{(1)}(N)(w^{(1)}(N))^*.$$

(9) Compute

$$d^{(1)}_B(k) = d_{U_1}(k+1) - p^{(1)}(k)(q^{(1)}(k))^*.$$

(10) *(Adjust indices).* Set

$$g_{V_1}(k) = \tilde{g}_{V_1}(k+1),\ k = 1,\dots,N, \quad h_{V_1}(k) = \tilde{h}_{V_1}(k+1),\ k = 2,\dots,N+1,$$

$$b_{V_1}(k) = \tilde{b}_{V_1}(k+1),\ k = 2,\dots,N;$$

$$d_{V_1}(1), d_{V_1}(N+1) \text{ to be the } 1\times 0 \text{ and } 0\times 1 \text{ empty matrices},$$

$$d_{V_1}(k) = \tilde{d}_{V_1}(k+1),\ k = 2,\dots,N$$

and

$$g_{U_1}(k) = \tilde{g}_{U_1}(k+1),\ k = 1,\dots,N,$$

$$h_{U_1}(k) = \tilde{h}_{U_1}(k+1),\ k = 2,\dots,N+1,$$

$$b_{U_1}(k) = \tilde{b}_{U_1}(k+1),\ k = 2,\dots,N,$$

$$d_{U_1}(k) = \tilde{d}_{U_1}(k+1),\ k = 1,\dots,N.$$

Although the formal proof of this algorithm is done via multiplication algorithms for matrices with quasiseparable representations (see [10, Theorem 31.4 and Theorem 36.4]), the explanation can also be given via the bulge-chasing process used in the standard QZ algorithms.

4.3. **Complexity.** A complexity estimate on the structured QZ algorithm above, applied to an $N \times N$ pencil and followed by the compression algorithm found in the Appendix, yields:

- $143 + 4\eta + (N-2)(251 + 7\eta)$ floating-point operations for the structured QZ update;
- $33 + (N-2)(3\eta + 144)$ operations for the compression of V;
- $3 + (N-2)(2\eta + 30)$ operations for the compression of U,

which gives a total count of $179 + 4\eta + (N-2)(425 + 12\eta)$ operations per iteration. Here η denotes the computational cost required to compute and apply a 2×2 Givens matrix.

5. Backward error analysis for companion pencils

This section is concerned with the backward error analysis of the structured QZ algorithm for companion pencils. The backward stability of structured QR/QZ algorithms based on generator representations has not received much attention in the literature so far. To our knowledge the only algorithm which is shown to be provably backward stable is the modification of the QR iteration for a rank-one correction of Hermitian matrices presented in [7].

This lack of works on stability analysis is probably due to the fact that generator computations are rather involved and do not admit a compact representation in terms of matrix manipulations. To circumvent this difficulty here we adopt the following hybrid point of view. We assume that the fast structured QR algorithm is backward stable. In this section based on a first order perturbation analysis we show that the initial perturbed problem has the same structure as the given problem and hence we derive the corresponding structured backward error expressed in terms of perturbations of the coefficients of the polynomial whose zeros are the computed roots. The results of a thorough numerical/experimental investigation are reported in the next section. These results confirm that the computed *a posteriori* bounds on the coefficients fit our analysis whenever we suppose that the backward error introduced by the fast variant of the QZ iteration is a small multiple of the same error for the customary algorithm. This is also in accordance with the conclusion stated in [7].

The problem of determining whether the QR method applied to the Frobenius companion matrix yields a backward stable root-finder is examined in [5], with a positive answer (Theorem 2.1) if the norm of the polynomial is small. Given a monic polynomial $p(x) = \sum_{j=0}^{n} a_j x^j$, let A be its companion matrix and E the perturbation matrix that measures the backward error of the QR method applied to A. Edelman and Murakami show that the coefficients of s^{k-1} in the polynomial $\det(sI_n - A - E) - \det(sI_n - A)$ are given at first order by the expression

$$\sum_{m=0}^{k-1} a_m \sum_{i=k+1}^{n} E_{i,i+m-k} - \sum_{m=k}^{n} a_m \sum_{i=1}^{k} E_{i,i+m-k}, \tag{5.1}$$

with $a_n = 1$.

A similar property holds for QZ applied to companion pencils. This result was proven in [19] by using a block elimination process, and later in [14] and [4] via a geometric approach. See also [15] for a backward error analysis that takes into account the effects of balancing.

In this section we rely on the results in [5] to derive explicit formulas for the polynomial backward error, at least for the case of nonsingular B. Here a_n is no longer necessarily equal to 1, although it is different from 0.

We consider companion pencils of the form

$$A = \begin{pmatrix} & & -a_0 \\ 1 & & -a_1 \\ & \ddots & \vdots \\ & 1 & -a_{n-1} \end{pmatrix}, \quad B = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ & & & a_n \end{pmatrix} \tag{5.2}$$

and we will denote the perturbation pencil as (E, G), where E and G are matrices of small norm.

First, observe that (5.1) can be generalized to the computation of $\det(s(I_n + G) - (A + E))$. Indeed, at first order we have

$$\begin{aligned} s(I_n + G) - (A + E) &= (I_n + G)[sI_n - (I_n + G)^{-1}(A + E)] \\ &= (I_n + G)[sI_n - (I_n - G)(A + E)] \\ &= (I_n + G)[sI_n - A - E + GA] \end{aligned}$$

and therefore

$$\det(s(I_n + G) - (A + E)) = \det(I_n + G)\det(sI_n - A - E + GA), \tag{5.3}$$

where (again at first order)

$$\det(I_n + G) = 1 + \operatorname{tr}(G) \tag{5.4}$$

and $\det(sI_n - A - E + GA)$ can be computed by applying (5.1) to the pencil $sI_n - A$ with the perturbation matrix $E - GA$. In order to fix the notation, let us write at first order

$$\det(sI_n - A - E + GA) = \det(sI_n - A) + \left(\sum_{j=0}^{n-1} (\Delta a)_j s^j \right). \tag{5.5}$$

Now, recall that B can be seen as a rank-one perturbation of the identity matrix: we can write $B = I_n + (a_n - 1)e_n e_n^T$, with the usual notation $e_n = [0, \ldots, 0, 1]^T$. So, the perturbed companion pencil is

$$s(B + G) - (A + E) = s(I_n + G) - (A + E) + s(a_n - 1)e_n e_n^T. \tag{5.6}$$

The Sherman-Morrison determinant formula applied to (5.6) gives

$$\begin{aligned} &\det(s(B + G) - (A + E)) \\ &= \det(s(I_n + G) - (A + E))(1 + s(a_n - 1)e_n^T[s(I_n + G) - (A + E)]^{-1}e_n). \end{aligned} \tag{5.7}$$

We know how to compute the determinant in the right-hand side of (5.7) thanks to the discussion above. Now we want to compute the second factor. What we need is the (n, n) entry of the matrix $[s(I_n + G) - (A + E)]^{-1}$, which can be written as

$$\begin{aligned} &[s(I_n + G) - (A + E)]^{-1}(n, n) \\ &= \det(s(I_n + G) - (A + E))^{-1} \cdot \det(s(I_{n-1} + \tilde{G}) - (\tilde{A} + \tilde{E})), \end{aligned} \tag{5.8}$$

where $\tilde{A} = A(1 : n-1, 1 : n-1)$, $\tilde{E} = E(1 : n-1, 1 : n-1)$ and $\tilde{G} = G(1 : n-1, 1 : n-1)$. Observe that $\tilde{A}$ is a companion matrix for the polynomial s^{n-1}, so, using

(5.3), we can write

$$\det(s(I_{n-1}+\tilde{G})-(\tilde{A}+\tilde{E}))=\det(I_{n-1}+\tilde{G})\left[s^{n-1}+\sum_{j=0}^{n-2}(\Delta\tilde{a})_j s^j\right],$$

where the coefficients $(\Delta\tilde{a})_j$ can be computed from (5.1) with the perturbation matrix $\tilde{E}-\tilde{G}\tilde{A}$.

From (5.3), (5.5), (5.7) and (5.8) we obtain

$$\begin{aligned}
&\det(s(B+G)-(A+E))\\
&=\det(s(I_n+G)-(A+E))+s(a_n-1)\det(s(I_{n-1}+\tilde{G})-(\tilde{A}+\tilde{E}))\\
&=\det(sI_n-A)+\sum_{j=0}^{n-1}(\Delta a)_j s^j+\operatorname{tr}(G)\det(sI_n-A)\\
&+s(a_n-1)\left[\sum_{j=0}^{n-2}(\Delta\tilde{a})_j s^j+(1+\operatorname{tr}(\tilde{G}))s^{n-1}\right]\\
&=p(s)+\operatorname{tr}(G)q(s)+\sum_{j=0}^{n-1}(\Delta a)_j s^j+s(a_n-1)\left[\sum_{j=0}^{n-2}(\Delta\tilde{a})_j s^j+\operatorname{tr}(\tilde{G})s^{n-1}\right],
\end{aligned}$$

where $q(x)=x^n+\sum_{j=0}^{n-1}a_j x^j$.

Finally, from $\det(s(B+G)-(A+E))$ we subtract the quantity $\det(sB-A)=p(s)$, and we obtain the following result:

Proposition 5.1. *With the above notation, the following equality is correct at first order:*

$$\begin{aligned}
&\det(s(B+G)-(A+E))-\det(sB-A)\\
&=\operatorname{tr}(G)q(s)+\sum_{j=0}^{n-1}(\Delta a)_j s^j+s(a_n-1)\left[\sum_{j=0}^{n-2}(\Delta\tilde{a})_j s^j+\operatorname{tr}(\tilde{G})s^{n-1}\right].
\end{aligned}\tag{5.9}$$

Remark 5.2. A first-order estimate of $\det(A+E-s(B+G))-\det(A-sB)$ can also be obtained via a geometric approach, as shown in [5] for companion matrices and in [14], [13] and [4] for pencils. Note that this approach is related to the Leverrier root-finding method: see [16] for an adaptation to the matrix pencil case.

6. Numerical results

In this section we test the performance of the fast QZ method presented above and compare it to classical unstructured QZ applied to the companion pencil and to classical QR applied to the companion matrix. We have implemented our method in Matlab and in Fortran 90/95; both implementations are available online.[2]

The first set of experiments are performed in Matlab: we use the commands `roots` for classical QR and `eig` for classical QZ, whereas fast QZ is applied using our Matlab implementation of the algorithm described above.

[2]http://www.unilim.fr/pages_perso/paola.boito/software.html

In particular, we report absolute forward and backward errors, measured in ∞-norm. For each polynomial $p(x)$ of degree N, the forward error is computed as

$$\text{forward error} = \max_{k=1,\ldots,N} \min_{j=1,\ldots,N} |\lambda_j - \alpha_k|,$$

where the λ_j's are the roots of $p(x)$ as determined by the eigensolver that is being studied, and the α_k's are the "exact" roots of $p(x)$, either known in advance or computed in high precision using Matlab's Symbolic Toolbox, unless otherwise specified. The backward error is computed as

$$\text{backward error} = \max_{k=0,\ldots,N} |\tilde{p}_k - p_k|,$$

where the p_k's are the exact coefficients of $p(x)$, either known in advance or computed in high precision from the known exact roots, and the $\tilde{p}_k$'s are the coefficients computed in high precision from the λ_j's.

Our aim here is to provide experimental evidence pointing to the stability of our structured QZ method and to the better accuracy of QZ versus QR on some classes of polynomials. We do not report timings for the various methods, since the running times for our Matlab implementation cannot be compared to the running times of a built-in function such as `eig` or `roots`.

Observe that the normalization of the polynomials is a crucial step for the proper functioning of QZ. Unlike the companion matrix, which is necessarily associated with a monic polynomial, the companion pencil allows for an arbitrary scaling of the polynomial. Unless otherwise specified, we normalize w.r.t. the 2-norm of the vector of coefficients. The polynomials obtained from computed roots, used for computation of backward errors, are also normalized in 2-norm. For the purpose of computing backward errors, we have also experimented with normalization in a least-squares sense, as suggested in [13], but in our examples we have generally found little difference between the 2-norm and the least-squares approach.

Example 6.1. Random polynomials.

We apply our structured QZ method to polynomials whose coefficients are random complex numbers with real and imaginary parts uniformly chosen in $[-1, 1]$. Here N denotes the degree. Such test polynomials are generally well conditioned and they are useful to study the behavior of our method as the polynomial degree grows larger.

Table 1 shows absolute forward errors w.r.t. the roots computed by the Matlab command `roots`, as well as the average number of iterations per eigenvalue, which is consistent with the expected number of operations for the classical QZ method. Backward errors (not shown in the table) are of the order of the machine epsilon. For each degree, errors and the number of iterations are averaged over 10 random polynomials.

Example 6.2. Cyclotomic polynomials $z^N - i$.

This is another set of well-conditioned polynomials for which we consider degrees up to 500. Table 2 shows forward errors with respect to the roots computed by the Matlab command `roots`, as well as the average number of iterations per eigenvalue. Backward errors (not shown in the table) are of the order of the machine epsilon.

TABLE 1. Errors and number of iterations for structured QZ applied to random polynomials; see Example 6.1.

N	abs. forward error	average n. iterations
50	2.60e−14	3.71
100	1.30e−13	3.59
150	4.52e−13	3.45
200	5.90e−13	3.38
300	1.69e−12	3.28
400	4.34e−12	3.22
500	6.11e−12	3.18

TABLE 2. Errors and number of iterations for structured QZ applied to cyclotomic polynomials; see Example 6.2.

N	abs. forward error	average n. iterations
50	1.01e−14	4.16
100	1.46e−14	4.01
150	3.55e−14	3.85
200	3.85e−14	3.88
300	7.23e−14	3.69
400	1.66e−13	3.66
500	1.78e−13	3.75

Example 6.3. We consider here a few examples taken from [5]:

(1) $p(x) = x^{20} + x^{19} + \cdots + x + 1$,
(2) the polynomial with roots equally spaced in the interval $[-2.1, 1.9]$,
(3) the Chebyshev polynomial of first kind of degree 20,
(4) the Bernoulli polynomial of degree 20.

Table 3 shows forward errors for fast and classical QZ (after normalization of the polynomial) and for classical balanced QR (before normalization), as well as the maximum eigenvalue condition number.

TABLE 3. Forward errors for polynomials in Example 6.3

polynomial	fast QZ	classical QZ	QR	max condeig
(1)	3.58e−15	1.09e−15	2.14e−15	1.38
(2)	7.87e−13	1.98e−13	5.24e−13	2.71e+4
(3)	4.16e−10	2.18e−11	3.62e−12	1.86e+296
(4)	4.00e−3	4.00e−3	4.00e−3	9.14e+6

Following [5], we also test our backward error analysis, given in Section 5, on these examples. Table 4 shows the logarithm in base 10, rounded to an entire number, of the computed and of the predicted backward errors for each coefficient of each polynomial. The predicted error is computed by applying (5.9).

As for the choice of E and G, we apply the backward error analysis given in [7], Theorem 4.1. This analysis implies that the backward error matrix for the

QR method applied to a small rank perturbation of a Hermitian $N \times N$ matrix M is, roughly speaking, bounded by a small multiple of $\varepsilon N^2 \|M\|_F$, where ε is the machine epsilon. In view of this result, each nonzero entry of E and G can be taken as a small multiple of $N\varepsilon$, where N is the degree of the polynomial. For the sparsity pattern of E and G, we follow the ideas in [5]. So, E and G are chosen here as `E=10*N*eps*triu(ones(N),-2)` and `G=10*N*eps*triu(ones(N),-1)`, with `N`=20, in order to model the backward error introduced by the QZ iterations. Note that, under these assumptions, the experimental results confirm the theoretical analysis.

TABLE 4. Backward errors for polynomials in Examples 6.3 and 6.4. Each entry of the table contains the logarithm in base 10 of the backward errors, in the following order: computed via fast QZ, computed via classical QZ, predicted according to (5.9).

	(1)	(2)	(3)	(4)	Ex. 6.4
z^{20}	$-14,-15,-13$	$-17,-17,-13$	$-16,-16,-13$	$-19,-19,-13$	$-26,-27,-13$
z^{19}	$-14,-15,-13$	$-17,-17,-15$	$-16,-16,-14$	$-18,-18,-16$	$-14,-15,-13$
z^{18}	$-14,-15,-13$	$-16,-16,-14$	$-15,-16,-13$	$-17,-18,-15$	$-15,-15,-14$
z^{17}	$-14,-15,-13$	$-16,-15,-14$	$-15,-16,-14$	$-17,-17,-16$	$-15,-15,-13$
z^{16}	$-15,-15,-13$	$-15,-16,-13$	$-15,-15,-12$	$-17,-17,-15$	$-15,-15,-13$
z^{15}	$-15,-15,-13$	$-15,-15,-14$	$-15,-15,-14$	$-16,-17,-15$	$-15,-15,-13$
z^{14}	$-14,-16,-13$	$-15,-15,-13$	$-15,-15,-12$	$-16,-16,-14$	$-15,-15,-13$
z^{13}	$-14,-15,-13$	$-15,-14,-14$	$-15,-16,-14$	$-15,-16,-15$	$-15,-15,-13$
z^{12}	$-15,-15,-13$	$-16,-15,-12$	$-15,-16,-12$	$-16,-16,-13$	$-15,-15,-13$
z^{11}	$-15,-15,-13$	$-15,-15,-14$	$-15,-15,-15$	$-15,-15,-14$	$-15,-15,-13$
z^{10}	$-15,-15,-13$	$-16,-16,-12$	$-15,-16,-13$	$-16,-16,-13$	$-15,-15,-13$
z^{9}	$-15,-15,-13$	$-15,-15,-14$	$-16,-16,-16$	$-14,-15,-14$	$-15,-15,-13$
z^{8}	$-15,-37,-13$	$-16,-15,-13$	$-16,-16,-14$	$-15,-16,-12$	$-15,-15,-13$
z^{7}	$-15,-15,-13$	$-15,-16,-15$	$-15,-16,-17$	$-14,-15,-14$	$-15,-15,-13$
z^{6}	$-14,-15,-13$	$-16,-15,-13$	$-15,-15,-15$	$-16,-40,-12$	$-15,-16,-13$
z^{5}	$-14,-16,-14$	$-16,-16,-16$	$-15,-16,-19$	$-14,-15,-14$	$-15,-15,-12$
z^{4}	$-14,-16,-14$	$-16,-16,-14$	$-15,-15,-16$	$-16,-16,-12$	$-15,-15,-13$
z^{3}	$-15,-15,-14$	$-17,-18,-17$	$-16,-16,-19$	$-15,-16,-17$	$-15,-14,-12$
z^{2}	$-15,-15,-28$	$-15,-17,-15$	$-15,-18,-18$	$-16,-17,-13$	$-15,-15,-13$
z^{1}	$-15,-15,-14$	$-18,-20,-17$	$-15,-\text{Inf},-19$	$-16,-16,-15$	$-15,-15,-12$
z^{0}	$-15,-15,-13$	$-17,-19,-13$	$-47,-\text{Inf},-13$	$-16,-17,-13$	$-15,-27,-13$

Example 6.4. QZ vs. QR.

In this example with heavily unbalanced coefficients, the (classical or structured) QZ method applied to the companion pencil computes the roots with better accuracy than QR applied to the companion matrix. We take here the polynomial $p = \sum_{k=0}^{20} p_k x^k$, where $p_k = 10^{6(-1)^{(k+1)}-3}$ for $k = 0, \ldots, 20$. Table 5 shows absolute forward and backward errors for several eigenvalue methods, with or without normalizing the polynomial before the computation of its roots. See Table 4 for backward errors.

We next show the results of experiments using the Fortran implementation of our method. The main goal of these experiments is to show that the structured method is actually faster than the classical LAPACK implementation for sufficiently high

TABLE 5. Errors for several versions of QZ and QR applied to a polynomial with unbalanced coefficients (Example 6.4).

Method	Backward error	Forward error
Fast QZ	2.45e−15	4.28e−15
Classical QZ	1.49e−15	3.22e−15
Balanced QR after normalization	1.22e−4	6.27e−9
Unbalanced QR after normalization	1.22e−4	1.72e−15
Balanced QR, no normalization	1.22e−4	5.86e−9
Unbalanced QR, no normalization	1.22e−4	2.72e−15

TABLE 6. Errors and running times (measured in seconds) for fast QZ implemented in Fortran versus the LAPACK implementation.

N	abs. forward error	fast QZ time	LAPACK time
50	1.73e−14	1.08e−2	7.10e−3
60	2.96e−14	1.66e−2	1.14e−2
70	3.32e−14	2.01e−2	1.76e−2
80	6.04e−14	2.43e−2	2.49e−2
90	7.74e−14	2.88e−2	3.51e−2
100	1.10e−13	3.26e−2	4.35e−2
150	1.60e−13	7.77e−2	1.35e−1
200	2.36e−13	1.29e−1	3.11e−1
300	8.44e−13	2.70e−1	9.91e−1
400	1.65e−12	4.66e−1	2.32
500	1.85e−12	7.27e−1	4.70

degrees, and that running times grow quadratically with the polynomial degree, as predicted by the complexity analysis. These experiments were performed on a MacBook Pro, using the `gfortran` compiler.

Example 6.5. Timings for random polynomials.

We take polynomials with random coefficients as in Example 6.1. Table 6 shows forward absolute errors with respect to the roots computed by LAPACK (subroutine ZGEGV), as well as timings (measured in seconds). Results for each degree are averaged over 10 trials.

Figure 1 shows running times versus polynomial degrees for Lapack and for our structured implementation. The crossover point for this experiment appears to be located at degree about 80.

Figure 2 shows a log-log plot of running times for our implementation, together with a linear fit. The slope of the fit is 1.9, which is consistent with the claim that the structured method has complexity $\mathcal{O}(n^2)$.

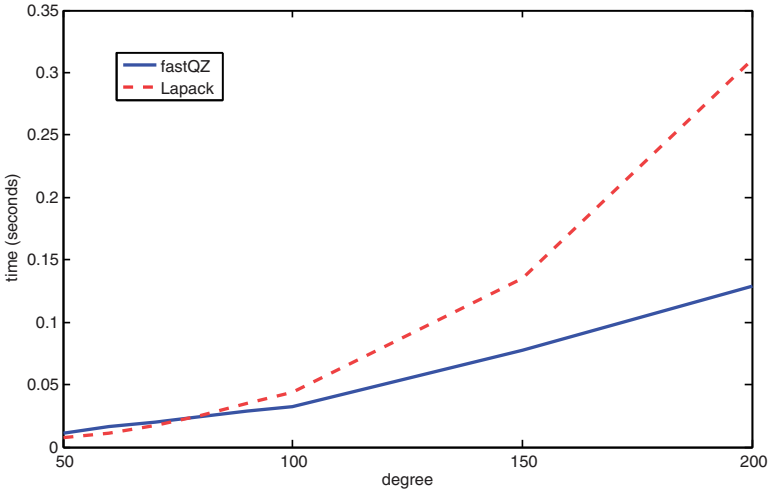

FIGURE 1. Timings for our fast QZ algorithm (blue solid line) and for LAPACK (red dashed line).

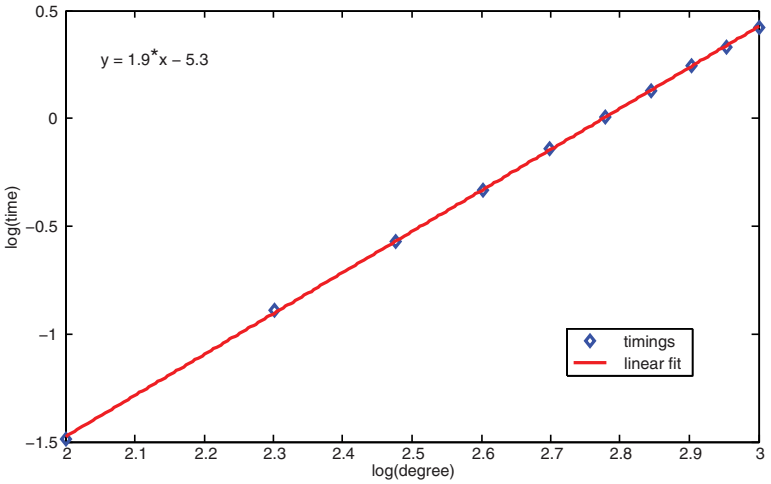

FIGURE 2. Linear fit on log-log plot of running times. Polynomial degrees go from 100 to 1000.

7. Conclusions

In this work we have presented and tested a new structured version of the single-shift, implicit QZ method. Our algorithm is designed for the fast computation of eigenvalues of matrix pencils belonging to a class $\mathcal{P}_N$ that includes companion and Lagrange pencils. The quasiseparable structure of such pencils allows us to achieve quadratic complexity.

Our numerical experience says that the fast structured implementation of the QZ algorithm applied to a companion pencil provides a fast and provably backward stable root-finding method.

Appendix A. The compression of generators

We present an algorithm that takes as input the possibly redundant quasiseparable generators of a unitary matrix and outputs generators of minimal order. This algorithm is justified in a similar way as the compression algorithm [9, Theorem 7.5] but with computations in the forward direction.

Let $U = \{U_{ij}\}_{i,j=1}^{N}$ be a block unitary matrix with entries of sizes $m_i \times n_j$, lower quasiseparable generators $p(i)$ $(i = 2, \dots, N)$, $q(j)$ $(j = 1, \dots, N-1)$, $a(k)$ $(k = 2, \dots, N-1)$ of orders r_k^L $(k = 1, \dots, N-1)$, upper quasiseparable generators $g(i)$ $(i = 1, \dots, N-1)$, $h(j)$ $(j = 2, \dots, N)$, $b(k)$ $(k = 2, \dots, N-1)$ of orders r_k^U $(k = 1, \dots, N-1)$ and diagonal entries $d(k)$ $(k = 1, \dots, N)$. Set

$$\begin{gathered} \rho_0 = 0,\ \rho_k = \min\{n_k + \rho_{k-1},\ r_k^L\},\ k = 1, \dots, N-1, \\ \nu_k = n_k + \rho_{k-1} - \rho_k,\ k = 1, \dots, N-1, \quad \nu_N = n_N + \rho_{N-1}, \\ s_0 = 0, \quad s_k = m_k + s_{k-1} - \nu_k, \quad k = 1, \dots, N-1. \end{gathered} \tag{A.1}$$

Then all the numbers s_k are nonnegative and the matrix U has upper quasiseparable generators of orders s_k $(k = 1, \dots, N-1)$. A set of such upper quasiseparable generators are obtained using the following algorithm.

Compression algorithm.

Input: lower quasiseparable generators $p(j)$, $q(j)$, $a(j)$ and upper quasiseparable generators $g(j)$, $h(j)$, $b(j)$ of possibly redundant orders for the matrix U.

Output: upper quasiseparable generators $g_s(j)$, $h_s(j)$, $b_s(j)$ of minimal order for U.

(1) Set X_0, Y_0, z_0 to be the 0×0 empty matrices and $p(1), a(1), h(1), b(1), h_s(1)$ to be empty matrices of sizes $m_1 \times 0, r_1^L \times 0, 0 \times n_1, 0 \times r_1^U, 0 \times n_1$ respectively.

(2) For $k = 1, \dots, N-1$ perform the following. Determine an $(n_k + \rho_{k-1}) \times (n_k + \rho_{k-1})$ unitary matrix W_k and an $r_k^L \times \rho_k$ matrix X_k such that

$$\begin{pmatrix} a(k)X_{k-1} & q(k) \end{pmatrix} W_k^* = \begin{pmatrix} 0_{r_k^L \times \nu_k} & X_k \end{pmatrix}.$$

W_k can be computed, for instance, via the usual Givens or Householder methods.

(3) Compute the $(m_k + s_{k-1}) \times (n_k + \rho_{k-1})$ matrix

$$Z_k = \begin{pmatrix} z_{k-1} & h_s(k) \\ p(k)X_{k-1}q(k) & d(k) \end{pmatrix} W_k^*.$$

(4) Determine the matrices $\Theta_k, \Delta_k, h_k', h_k''$ of sizes $s_{k-1} \times \nu_k, m_k \times \nu_k, s_{k-1} \times \rho_k, m_k \times \rho_k$ from the partition

$$Z_k = \begin{bmatrix} \Theta_k & h_k' \\ \Delta_k & h_k'' \end{bmatrix}.$$

Observe that the submatrix $\begin{pmatrix} \Theta_k \\ \Delta_k \end{pmatrix}$ has orthonormal columns.

(5) Determine an $(s_{k-1} + m_k) \times (s_{k-1} + m_k)$ unitary matrix F_k from the condition

$$F_k^* \begin{pmatrix} \Theta_k \\ \Delta_k \end{pmatrix} = \begin{pmatrix} I_{\nu_k} \\ 0_{s_k \times \nu_k} \end{pmatrix}.$$

(6) Determine the matrices $h_F(k), d_F(k), b_s(k), g_s(k)$ of sizes $s_{k-1} \times \nu_k, m_k \times \nu_k, s_{k-1} \times s_k, m_k \times s_k$ from the partition

$$F_k = \begin{bmatrix} h_F(k) & b_s(k) \\ d_F(k) & g_s(k) \end{bmatrix}.$$

(7) Compute the matrices Y_k of size $s_k \times r_k^U$ and z_k of the size $s_k \times \rho_k$ by the formulas

$$Y_k = g_s^*(k)g(k) + b_s^*(k)Y_{k-1}b(k), \quad z_k = g_s^*(k)h_k'' + b_s^*(k)h_k'.$$

(8) Compute

$$h_s(k+1) = Y_k h(k+1).$$

(9) Set

$$F_N = \begin{bmatrix} z_{N-1} & Y_{N-1}h(N) \\ p(N)X_{N-1} & d(N) \end{bmatrix}.$$

For a matrix from the class $\mathcal{U}_N$ we have $m_i = n_i = 1,\ i = 1, \ldots, N$ and $r_k^L = 1$, $k = 1, \ldots, N-1$ which by virtue of (A.1) implies $\rho_k = s_k = 1,\ k = 1, \ldots, N-1$.

For a matrix from the class $\mathcal{V}_N$ we determine the sizes of blocks via (3.2) and the orders of lower generators $r_k^L = 1,\ k = 1, \ldots, N$. Hence using (A.1) we obtain $s_1 = 1,\ s_k = 2,\ k = 2, \ldots, N-1$.

Remark A.1. With the above hypotheses, the matrix U admits the (nonunique) factorization

$$U = W \cdot F,$$

where W is a block lower triangular unitary matrix with block entries of sizes $m_i \times \nu_j$ $(i, j = 1, \ldots, N)$ and F is a block upper triangular unitary matrix with block entries of sizes $\nu_i \times n_j$ $(i, j = 1, \ldots, N)$. Moreover, one can choose the matrix W in the form

$$W = \tilde{W}_{N-1} \cdots \tilde{W}_1, \quad F = \tilde{F}_1 \cdots \tilde{F}_N$$

with

$$\tilde{W}_k = \operatorname{diag}\{I_{\phi_k}, W_k, I_{\eta_k}\},\ k = 1, \ldots, N-1;\ \tilde{F}_k = \operatorname{diag}\{I_{\phi_k}, F_k, I_{\chi_k}\},\ k = 1, \ldots, N,$$

for suitable sizes ϕ_k, η_k and χ_k. Here W_k $(k = 1, \ldots, N-1)$ are $(n_k + \rho_{k-1}) \times (n_k + \rho_{k-1})$ and F_k $(k = 1, \ldots, N)$ are $(s_{k-1} + m_k) \times (s_{k-1} + m_k)$ unitary matrices.

References

[1] V. I. Arnol′d, *Matrices depending on parameters* (Russian), Uspehi Mat. Nauk **26** (1971), no. 2(158), 101–114. MR0301242 (46 #400)

[2] D. Day and L. Romero, *Roots of polynomials expressed in terms of orthogonal polynomials*, SIAM J. Numer. Anal. **43** (2005), no. 5, 1969–1987 (electronic), DOI 10.1137/040609847. MR2192327 (2006h:65070)

[3] C. de Boor, *An empty exercise*, in ACM SIGNUM Newsletter, vol. 25 (4), Oct. 1990, pp. 2–6.

[4] A. Edelman, E. Elmroth, and B. Kågström, *A geometric approach to perturbation theory of matrices and matrix pencils. I. Versal deformations*, SIAM J. Matrix Anal. Appl. **18** (1997), no. 3, 653–692, DOI 10.1137/S0895479895284634. MR1453545 (99e:58021)

[5] A. Edelman and H. Murakami, *Polynomial roots from companion matrix eigenvalues*, Math. Comp. **64** (1995), no. 210, 763–776, DOI 10.2307/2153450. MR1262279 (95f:65075)

[6] Y. Eidelman, I. Gohberg, and L. Gemignani, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra Appl. **420** (2007), no. 1, 86–101, DOI 10.1016/j.laa.2006.06.028. MR2277631 (2007j:15011)

[7] Y. Eidelman, L. Gemignani, and I. Gohberg, *Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations*, Numer. Algorithms **47** (2008), no. 3, 253–273, DOI 10.1007/s11075-008-9172-0. MR2385737 (2009a:65090)

[8] Y. Eidelman and I. Gohberg, *On a new class of structured matrices*, Integral Equations Operator Theory **34** (1999), no. 3, 293–324, DOI 10.1007/BF01300581. MR1689391 (2000e:15020)

[9] Y. Eidelman, I. Gohberg, and I. Haimovici, *Separable Type Representations of Matrices and Fast Algorithms. Vol. 1*: *Basics. Completion Problems. Multiplication and Inversion Algorithms*, Operator Theory: Advances and Applications, vol. 234, Birkhäuser/Springer, Basel, 2014. MR3137083

[10] Y. Eidelman, I. Gohberg, and I. Haimovici, *Separable Type Representations of Matrices and Fast Algorithms. Vol. 2*: *Eigenvalue Method*, Operator Theory: Advances and Applications, vol. 235, Birkhäuser/Springer Basel AG, Basel, 2014. MR3136431

[11] M. Fiedler and T. L. Markham, *Completing a matrix when certain entries of its inverse are specified*, Linear Algebra Appl. **74** (1986), 225–237, DOI 10.1016/0024-3795(86)90125-4. MR822149 (87g:15005)

[12] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4th ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2013. MR3024913

[13] G. F. Jónsson and S. Vavasis, *Solving polynomials with small leading coefficients*, SIAM J. Matrix Anal. Appl. **26** (2004/05), no. 2, 400–414, DOI 10.1137/S0895479899365720. MR2124155 (2005k:12010)

[14] D. Lemonnier, P. Van Dooren, *Optimal Scaling of Companion Pencils for the QZ-Algorithm*, SIAM Conference on Applied Linear Algebra, Williamsburg, VA. 2003.

[15] D. Lemonnier and P. Van Dooren, *Balancing regular matrix pencils*, SIAM J. Matrix Anal. Appl. **28** (2006), no. 1, 253–263, DOI 10.1137/S0895479804440931. MR2218952 (2007a:15014)

[16] B. G. Mertzios, *Leverrier's algorithm for singular systems*, IEEE Trans. Automat. Control **29** (1984), no. 7, 652–653, DOI 10.1109/TAC.1984.1103602. MR748757

[17] M. Sebek, H. Kwakernaak, D. Henrion and S. Pejchova, *Recent progress in polynomial methods and Polynomial Toolbox for Matlab version 2.0*, Proc. of the 37th IEEE Conference on Decision and Control, 1998, vol. 4, pp. 3661–3668.

[18] K.-C. Toh and L. N. Trefethen, *Pseudozeros of polynomials and pseudospectra of companion matrices*, Numer. Math. **68** (1994), no. 3, 403–425, DOI 10.1007/s002110050069. MR1313152 (95m:65085)

[19] P. Van Dooren and P. Dewilde, *The eigenstructure of an arbitrary polynomial matrix: computational aspects*, Linear Algebra Appl. **50** (1983), 545–579, DOI 10.1016/0024-3795(83)90069-1. MR699575 (84j:15009)

[20] D. S. Watkins, *Fundamentals of Matrix Computations*, Pure and Applied Mathematics (New York), Wiley-Interscience [John Wiley & Sons], New York, 2002. Second editon. MR1899577 (2003a:65002)

XLIM–DMI, UMR CNRS 7252, Faculté des Sciences et Techniques, 123 av. A. Thomas, 87060 Limoges, France

E-mail address: paola.boito@unilim.fr

School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Ramat-Aviv, 69978, Israel

E-mail address: eideyu@post.tau.ac.il

Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa, Italy

E-mail address: l.gemignani@di.unipi.it