



A Real QZ Algorithm for Structured Companion Pencils

Paola Boito, Yuli Eidelman, Luca Gemignani

► **To cite this version:**

Paola Boito, Yuli Eidelman, Luca Gemignani. A Real QZ Algorithm for Structured Companion Pencils. *Calcolo*, Springer Verlag, 2017, <10.1007/s10092-017-0231-6>. <hal-01407864>

HAL Id: hal-01407864

<https://hal.inria.fr/hal-01407864>

Submitted on 2 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Real QZ Algorithm for Structured Companion Pencils

P. Boito · Y. Eidelman · L. Gemignani

Received: date / Accepted: date

Abstract We design a fast implicit real QZ algorithm for eigenvalue computation of structured companion pencils arising from linearizations of polynomial rootfinding problems. The modified QZ algorithm computes the generalized eigenvalues of an $N \times N$ structured matrix pencil using $O(N^2)$ flops and $O(N)$ memory storage. Numerical experiments and comparisons confirm the effectiveness and the stability of the proposed method.

Keywords Rank-structured matrix · Quasiseparable matrix · Real QZ algorithm · Lagrange approximation · eigenvalue computation · complexity

Mathematics Subject Classification (2000) 65F15 · 65H17

1 Introduction

Linearization techniques based on polynomial interpolation are becoming nowadays a standard way to solve numerically nonlinear zerofinding problems for polynomials or more generally for analytic functions [1]. Since in many applications the interest is in the approximation of real zeros, methods using Chebyshev-like expansions are usually employed. Alternatively, Lagrange interpolation at the roots of unity can be considered. For a real function a straightforward modification of

This work was partially supported by GNCS-INDAM and University of Pisa.

P. Boito
XLIM-DMI UMR CNRS 7252 Faculté des Sciences et Techniques, 123 avenue A. Thomas,
87060 Limoges, France
E-mail: paola.boito@unilim.fr

Y. Eidelman
School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences,
Tel-Aviv University, Ramat-Aviv, 69978, Israel
E-mail: eideyu@post.tau.ac.il

L. Gemignani
Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
E-mail: l.gemignani@di.unipi.it

the classical approach [5, 15] yields a structured companion pencil $\mathcal{A}(\lambda) = F - \lambda G$ where F and G are real $N \times N$ low rank corrections of unitary matrices. The computation of the generalized eigenvalues of this pencil can be performed by means of the QZ algorithm [19] suitably adjusted to work in real arithmetic.

In this paper we propose a fast adaptation of the real QZ algorithm for computing the generalized eigenvalues of certain $N \times N$ structured pencils using only $O(N^2)$ flops and $O(N)$ memory storage. The pencils $\mathcal{A}(\lambda) = F - \lambda G$, $F, G \in \mathbb{R}^{N \times N}$, we consider here satisfy two basic properties:

1. F is upper Hessenberg and G is upper triangular;
2. F and G are rank-one corrections of unitary matrices.

We refer to such a pencil $\mathcal{A}(\lambda)$ as a companion-like pencil, since the class includes companion pencils as a special case. Sometimes $\mathcal{A}(\lambda)$ is also denoted by $(F, G) \in \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times N}$.

Let (F_k, G_k) , $k \geq 0$, $F_0 = F, G_0 = G$, be the sequence of matrix pairs (pencils) generated by the real QZ algorithm starting from the companion-like pencil $\mathcal{A}(\lambda)$. Single or double shifting is applied in the generic iteration $F_k \rightarrow F_{k+1}$, $G_k \rightarrow G_{k+1}$ in order to carry out all the computations in real arithmetic. Whatever strategy is used, it is found that both $\mathcal{A}_k(\lambda)$ and $\mathcal{A}_{k+1}(\lambda)$ are still companion-like pencils. As a consequence of this invariance we obtain that all the matrices involved in the QZ iteration inherit a rank structure in their upper triangular parts. This makes it possible to represent F_k, G_k and F_{k+1}, G_{k+1} as data-sparse matrices specified by a number of parameters (called generators) which is linear w.r.t. the size of the matrices.

In this paper we introduce a convenient set of generators and design a structured variant of the real QZ iteration which takes in input the generators of F_k and G_k together with the shift parameters and returns as output the generators of F_{k+1} and G_{k+1} . It is shown that the arithmetic cost for each iteration is $O(N)$ using linear memory storage. Numerical experiments confirm the effectiveness and the robustness of the resulting eigensolver.

The paper is organized as follows. In Section 2 we set up the scene by introducing the matrix problem and its basic properties. In Section 3 we define an appropriate set of generators for the matrices involved. In Section 4 we design the fast adaptation of the QZ algorithm using these generators and exploiting the resulting data-sparse representations. We focus here on double shifting, since the single shifted iteration has been already devised in [4]. A proof of the correctness of the algorithm is given in Appendix. Finally, in Section 5 we show the results of numerical experiments, whereas conclusion and future work are presented in Section 6.

2 The Problem Statement

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

Definition 1 The matrix pair (A, B) , $A, B \in \mathbb{R}^{N \times N}$, belongs to the class $\mathcal{P}_N \subset \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times N}$ of companion-like pencils iff:

1. $A \in \mathbb{R}^{N \times N}$ is upper Hessenberg;

2. $B \in \mathbb{R}^{N \times N}$ is upper triangular;
 3. There exist two vectors $\mathbf{z} \in \mathbb{R}^N$ and $\mathbf{w} \in \mathbb{R}^N$ and an orthogonal matrix $V \in \mathbb{R}^{N \times N}$ such that

$$A = V - \mathbf{z}\mathbf{w}^*; \quad (2.1)$$

4. There exist two vectors $\mathbf{p} \in \mathbb{R}^N$ and $\mathbf{q} \in \mathbb{R}^N$ and an orthogonal matrix $U \in \mathbb{R}^{N \times N}$ such that

$$B = U - \mathbf{p}\mathbf{q}^*. \quad (2.2)$$

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

Definition 2 We denote by \mathcal{T}_N the class of upper triangular matrices $B \in \mathbb{R}^{N \times N}$ which are rank-one perturbations of orthogonal matrices, i.e., such that (2.2) holds for a suitable orthogonal matrix U and vectors \mathbf{p}, \mathbf{q} .

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

$$U(i, j) = p(i)q^*(j), \quad 1 \leq j < i \leq N, \quad (2.3)$$

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

Definition 3 We denote by \mathcal{U}_N the class of orthogonal matrices $U \in \mathbb{R}^{N \times N}$ which satisfy the condition (2.3), i.e., for which there exist vectors \mathbf{p}, \mathbf{q} such that the matrix $B = U - \mathbf{p}\mathbf{q}^*$ is an upper triangular matrix.

Observe that we have

$$U \in \mathcal{U}_N \Rightarrow \text{rank } U(k+1: N, 1: k) \leq 1, \quad k = 1, \dots, N-1.$$

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,

$$U \in \mathcal{U}_N \Rightarrow \text{rank } U(1: k, k+1: N) \leq 1, \quad k = 1, \dots, N-1. \quad (2.4)$$

Definition 4 We denote by \mathcal{H}_N the class of upper Hessenberg matrices $A \in \mathbb{R}^{N \times N}$ which are rank one perturbations of orthogonal matrices, i.e., such that (2.1) holds for a suitable orthogonal matrix V and vectors \mathbf{z}, \mathbf{w} .

Definition 5 We denote by \mathcal{V}_N the class of orthogonal matrices $V \in \mathbb{R}^{N \times N}$ for which there exist vectors \mathbf{z}, \mathbf{w} such that the matrix $A = V - \mathbf{z}\mathbf{w}^*$ is an upper Hessenberg matrix.

We find that

$$V \in \mathcal{V}_N \Rightarrow \text{rank } V(k+2: N, 1: k) \leq 1, \quad k = 1, \dots, N-2.$$

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

$$V \in \mathcal{V}_N \Rightarrow \text{rank } V(1: k, k: N) \leq 2, \quad k = 1, \dots, N. \quad (2.5)$$

In this paper we consider the problem of efficiently computing the (generalized) eigenvalues of a companion-like matrix pencil $(A, B) \in \mathcal{P}_N$ by exploiting the rank and banded structures of the matrix classes mentioned above. The QZ algorithm is the customary method for solving generalized eigenvalue problems numerically by means of unitary transformations (see e.g. [12] and [19]). For the pair (A, B) in Hessenberg/triangular form the implicit QZ step consists in the computation of unitary matrices Q and Z such that

$$A_1 = Q^*AZ \text{ is upper Hessenberg, } B_1 = Q^*BZ \text{ is upper triangular} \quad (2.6)$$

and some initial conditions hold. For the QZ iteration applied to a real matrix pair with double shifting the initial condition is

$$(Q^*p(AB^{-1}))(:, 2:N) = 0, \quad (2.7)$$

where $p(z) = \alpha + \beta z + \gamma z^2$ is the shift polynomial. In this case one obtains the orthogonal Hessenberg matrices Q and Z in the form

$$Q = \tilde{Q}_1 \tilde{Q}_2 \cdots \tilde{Q}_{N-2} \tilde{Q}_{N-1}, \quad Z = \tilde{Z}_1 \tilde{Z}_2 \cdots \tilde{Z}_{N-2} \tilde{Z}_{N-1}, \quad (2.8)$$

where

$$\begin{aligned} \tilde{Q}_i &= I_{i-1} \oplus Q_i \oplus I_{N-i-2}, \quad i = 1, \dots, N-2, & \tilde{Q}_{N-1} &= I_{N-2} \oplus Q_{N-1}, \\ \tilde{Z}_i &= I_{i-1} \oplus Z_i \oplus I_{N-i-2}, \quad i = 1, \dots, N-2, & \tilde{Z}_{N-1} &= I_{N-2} \oplus Z_{N-1} \end{aligned} \quad (2.9)$$

and Q_i, Z_i , $i = 1, \dots, N-2$ are 3×3 orthogonal matrices, Q_{N-1}, Z_{N-1} are real Givens rotations.

Since the Hessenberg/triangular form is preserved under the QZ iteration an easy computation then yields

$$(A, B) \in \mathcal{P}_N, (A, B) \xrightarrow{\text{QZ step}} (A_1, B_1) \Rightarrow (A_1, B_1) \in \mathcal{P}_N. \quad (2.10)$$

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

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

with the unitary matrices $V_1 = Q^*VZ$, $U_1 = Q^*UZ$ and the vectors $\mathbf{z}_1 = Q^*z$, $\mathbf{w}_1 = Z^*w$, $\mathbf{p}_1 = Q^*p$, $\mathbf{q}_1 = Z^*q$. 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 exploit 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 $(A, B) \in \mathcal{P}_N$.

A matrix $M = \{M_{ij}\}_{i,j=1}^N$ is (r^L, r^U) -*quasiseparable*, with r^L, r^U positive integers, if, using MATLAB¹ notation,

$$\begin{aligned} \max_{1 \leq k \leq N-1} \text{rank}(M(k+1:N, 1:k)) &\leq r^L, \\ \max_{1 \leq k \leq N-1} \text{rank}(M(1:k, k+1:N)) &\leq r^U. \end{aligned}$$

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 section 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 and clarity, generating matrices and vectors are denoted by a roman lower-case letter.

In this representation, the entries of M take the form

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

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

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

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

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.

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 ones by choosing block sizes as

$$m_1 = \dots = m_N = 1, m_{N+1} = 0, \quad n_1 = 0, n_2 = \dots = n_{N+1} = 1. \quad (3.2)$$

Such a treatment allows also 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

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

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 [6]. In particular, the product of a $m \times 0$ matrix by a $0 \times m$ matrix is a $m \times m$ 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.

3.1 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 $\mathbf{z} = \text{col}(z(i))_{i=1}^N$, $\mathbf{w} = \text{col}(w(i))_{i=1}^N$.

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 $\mathbf{p} = \text{col}(p(i))_{i=1}^N$, $\mathbf{q} = \text{col}(q(i))_{i=1}^N$.

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

All the given parameters define completely the matrix pair (A, B) from the class \mathcal{P}_N . Updating of these parameters while keeping the minimal orders of generators is a task of the fast QZ iteration described in the next section.

4 A fast implicit double shifted QZ iteration via generators

In this section we present our fast adaptation of the double-shifted QZ algorithm for a matrix pair $(A, B) \in \mathcal{P}_N$. The algorithm takes in input a quasiseparable representation of the matrices A and B together with the coefficients of the real quadratic shift polynomial and it returns as output a possibly not minimal quasiseparable representation of the matrices $(A_1, B_1) \in \mathcal{P}_N$ such that (2.10) holds. The algorithm computes the unitary matrices Q_i and Z_i defined in (2.9). It basically splits into the following four stages:

1. a **preparative phase** where Q_1 is found so as to satisfy the shifting condition;
2. the **chasing the bulge** step where the unitary matrices Q_2, \dots, Q_{N-2} and Z_1, \dots, Z_{N-3} are determined in such a way to perform the Hessenberg/triangular reduction procedure;
3. a **closing** phase where the last three transformations Q_{N-1} , Z_{N-2} and Z_{N-1} are carried out;
4. the final stage of **recovering the generators** of the updated pair.

For the sake of brevity the stage 2 and 3 are grouped together by using empty and zero quantities when needed. The correctness of the algorithm is proved in the Appendix. Some technical details concerning shifting strategies and shifting techniques are discussed in the section on numerical experiments. Compression of generators yielding minimal representations can be achieved by using the methods devised in [4]. The incorporation of these compression schemes does not alter the complexity of the main algorithm shown below.

ALGORITHM: Implicit QZ iteration for companion-like pencils with double shift

1. **INPUT:**

- (a) the subdiagonal entries σ_k^A ($k = 1, \dots, N-1$) of the matrix A ;
- (b) the upper triangular generators $g_V(i), h_V(i)$ ($i = 1, \dots, N$), $b_V(k)$ ($k = 1, \dots, N-1$) with orders r_k^V ($k = 1, \dots, N$) of the matrix V ;

- (c) the diagonal entries $d_B(k)$ ($k = 1, \dots, N$) of the matrix B ;
- (d) the upper quasiseparable generators $g_U(i)$ ($i = 1, \dots, N-1$), $h_U(j)$ ($j = 2, \dots, N$), $b_U(k)$ ($k = 2, \dots, N-1$) with orders r_k^U ($k = 1, \dots, N-1$) of the matrix U ;
- (e) the perturbation vectors $\mathbf{z} = \text{col}(z(i))_{i=1}^N$, $\mathbf{w} = \text{col}(w(i))_{i=1}^N$, $\mathbf{p} = \text{col}(p(i))_{i=1}^N$, $\mathbf{q} = \text{col}(q(i))_{i=1}^N$;
- (f) the coefficients of the shift polynomial $p(z) = \alpha + \beta z + \gamma z^2 \in \mathbb{R}[z]$;
2. **OUTPUT:**
- (a) the subdiagonal entries $\sigma_k^{A_1}$ ($k = 1, \dots, N-1$) of the matrix A_1 ;
- (b) upper triangular generators $g_V^{(1)}(i), h_V^{(1)}(i)$ ($i = 1, \dots, N$), $b_V^{(1)}(k)$ ($k = 1, \dots, N-1$) of the matrix V_1 ;
- (c) the diagonal entries $d_B^{(1)}(k)$ ($k = 1, \dots, N$) of the matrix B_1 ;
- (d) upper quasiseparable generators $g_U^{(1)}(i)$ ($i = 1, \dots, N-1$), $h_U^{(1)}(j)$ ($j = 2, \dots, N$), $b_U^{(1)}(k)$ ($k = 2, \dots, N-1$) of the matrix U_1 ;
- (e) perturbation vectors $\mathbf{z}_1 = \text{col}(z^{(1)}(i))_{i=1}^N$, $\mathbf{w}_1 = \text{col}(w^{(1)}(i))_{i=1}^N$, $\mathbf{p}_1 = \text{col}(p^{(1)}(i))_{i=1}^N$, $\mathbf{q}_1 = \text{col}(q^{(1)}(i))_{i=1}^N$;
3. **COMPUTATION:**

– **Preparative Phase**

- (a) Compute $\mathbf{s} = (p(AB^{-1})\mathbf{e}_1)(1:3)$ and determine the 3×3 orthogonal matrix Q_1 from the condition

$$Q_1^* \mathbf{s} = (* 0 0)^*. \quad (4.1)$$

- (b) Compute

$$\begin{pmatrix} \tilde{g}_V^{(3)} \\ \beta_3^V \end{pmatrix} = Q_1^* \begin{pmatrix} g_V(1)h_V(1) & g_V(1)b_V(1)h_V(2) & g_V(1)b_V(1)b_V(2) \\ \sigma_1^V & g_V(2)h_V(2) & g_V(2)b_V(2) \\ z(3)w(1) & \sigma_2^V & g_V(3) \end{pmatrix} \quad (4.2)$$

and determine the matrices f_3^V, ϕ_3^V of sizes $2 \times 2, 2 \times r_3^V$ from the partition

$$\beta_3^V = [f_3^V \ \phi_3^V]. \quad (4.3)$$

- (c) Compute

$$\begin{pmatrix} z^{(1)}(1) \\ \chi_3 \end{pmatrix} = Q_1^* \begin{pmatrix} z(1) \\ z(2) \\ z(3) \end{pmatrix}, \quad \gamma_2 = \begin{pmatrix} w(1) \\ w(2) \end{pmatrix} \quad (4.4)$$

with the number $z^{(1)}$ and two-dimensional columns χ_3, γ_2 . Compute

$$f_3^A = f_3^V - \chi_3 \gamma_2^*, \quad \phi_3^A = \phi_3^V. \quad (4.5)$$

- (d) Set

$$c_2 = \begin{pmatrix} p(1) \\ p(2) \end{pmatrix}, \quad \theta_1 = q(1), \quad \theta_2 = \begin{pmatrix} q(1) \\ q(2) \end{pmatrix}. \quad (4.6)$$

Compute

$$d_U(1) = d_B(1) + p(1)q(1), \quad d_U(2) = d_B(2) + p(2)q(2) \quad (4.7)$$

and set

$$f_2^U = \begin{pmatrix} d_U(1) & g_U(1)h_U(2) \\ p(2)q(1) & d_U(2) \end{pmatrix}, \quad \phi_2^U = \begin{pmatrix} g_U(1)b_U(2) \\ g_U(2) \end{pmatrix}, \quad (4.8)$$

$$\varepsilon = g_U(1)h_U(2) - p(1)q(2), \quad (4.9)$$

$$f_2^B = \begin{pmatrix} d_B(1) & \varepsilon \\ 0 & d_B(2) \end{pmatrix}, \quad \varphi_2^B = \phi_2^U. \quad (4.10)$$

– **Chasing the Bulge** For $k = 1, \dots, N - 1$ perform the following:

(a) (*Apply Q_k and determine Z_k*). Compute the two-dimensional column ε_{k+1}^B via

$$\varepsilon_{k+1}^B = \varphi_{k+1}^B h_U(k+2) - c_{k+1} q(k+2), \quad (4.11)$$

and the 3×3 matrix Φ_k by the formula

$$\Phi_k = Q_k^* \begin{pmatrix} f_{k+1}^B & \varepsilon_{k+1}^B \\ 0 & d_B(k+2) \end{pmatrix}. \quad (4.12)$$

Determine the 3×3 orthogonal matrix Z_k such that

$$\Phi_k(2 : 3, :) Z_k = \begin{pmatrix} 0 & * & * \\ 0 & 0 & * \end{pmatrix}. \quad (4.13)$$

(b) (*Determine Q_{k+1}*). Compute the column

$$\varepsilon_{k+2}^A = \varphi_{k+2}^A h_V(k+2) - \chi_{k+2} w(k+2) \quad (4.14)$$

and the 3×3 matrix Ω_k by the formula

$$\Omega_k = \begin{pmatrix} f_{k+2}^A & \varepsilon_{k+2}^A \\ 0 & \sigma_{k+2}^A \end{pmatrix} Z_k. \quad (4.15)$$

Determine the 3×3 orthogonal matrix Q_{k+1} and the number $(\sigma_k^A)^{(1)}$ such that

$$Q_{k+1}^* \Omega_k(:, 1) = \begin{pmatrix} (\sigma_k^A)^{(1)} \\ 0 \\ 0 \end{pmatrix}. \quad (4.16)$$

(c) (*Update generators for U and B*). Compute

$$d_U(k+2) = d_B(k+2) + p(k+2)q(k+2), \quad (4.17)$$

$$\begin{pmatrix} \tilde{d}_U(k+2) & \tilde{g}_U(k+2) \\ * & \beta_{k+2}^U \end{pmatrix} = Q_k^* \tilde{U}_k \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+2}^U} \end{pmatrix} \quad (4.18)$$

where

$$\tilde{U}_k = \begin{pmatrix} f_{k+1}^U & \phi_{k+1}^U h_U(k+2) & \phi_{k+1}^U b_U(k+2) \\ p(k+2)\theta_{k+1}^* & d_U(k+2) & g_U(k+2) \end{pmatrix}$$

and determine the matrices f_{k+2}^U, ϕ_{k+2}^U of sizes $2 \times 2, 2 \times r_{k+2}^U$ from the partition

$$\beta_{k+2}^U = [f_{k+2}^U \quad \phi_{k+2}^U]. \quad (4.19)$$

Compute

$$\begin{aligned} & (\tilde{h}_U(k+2) \tilde{b}_U(k+2)) = \\ & \begin{pmatrix} I_2 & 0 & 0 \\ 0 & h_U(k+2) & b_U(k+2) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+2}^U} \end{pmatrix}. \end{aligned} \quad (4.20)$$

Compute

$$\begin{aligned} \begin{pmatrix} p^{(1)}(k) \\ c_{k+2} \end{pmatrix} &= Q_k^* \begin{pmatrix} c_{k+1} \\ p(k+2) \end{pmatrix} \\ \begin{pmatrix} q^{(1)}(k) \\ \theta_{k+2} \end{pmatrix} &= Z_k^* \begin{pmatrix} \theta_{k+1} \\ q(k+2) \end{pmatrix} \end{aligned} \quad (4.21)$$

with the numbers $p^{(1)}(k), q^{(1)}(k)$ and two-dimensional columns c_{k+2}, θ_{k+2} .

Compute

$$f_{k+2}^B = f_{k+2}^U - c_{k+2} \theta_{k+2}^*, \quad \varphi_{k+2}^B = \phi_{k+2}^U. \quad (4.22)$$

(d) (*Update generators for V and A*). Compute

$$\sigma_{k+2}^V = \sigma_{k+2}^A + z(k+3)w(k+2), \quad (4.23)$$

$$\begin{aligned} & \begin{pmatrix} \tilde{d}_V(k+3) & \tilde{g}_V(k+3) \\ * & \beta_{k+3}^V \end{pmatrix} = \\ & Q_{k+1}^* \tilde{V}_{k+2} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+3}^V} \end{pmatrix}, \end{aligned} \quad (4.24)$$

where

$$\tilde{V}_{k+2} = \begin{pmatrix} f_{k+2}^V & \phi_{k+2}^V h_V(k+2) & \phi_{k+2}^V b_V(k+2) \\ z(k+3)\gamma_{k+1}^* & \sigma_{k+2}^V & g_V(k+3) \end{pmatrix}.$$

Determine the matrices f_{k+3}^V, ϕ_{k+3}^V of sizes $2 \times 2, 2 \times r_{k+3}^V$ from the partition

$$\beta_{k+3}^V = [f_{k+3}^V \ \phi_{k+3}^V]. \quad (4.25)$$

Compute

$$\begin{aligned} & (\tilde{h}_V(k+3) \tilde{b}_V(k+3)) = \\ & \begin{pmatrix} I_2 & 0 & 0 \\ 0 & h_V(k+2) & b_V(k+2) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+3}^V} \end{pmatrix}, \end{aligned} \quad (4.26)$$

and

$$\begin{aligned} \begin{pmatrix} z^{(1)}(k+1) \\ \chi_{k+3} \end{pmatrix} &= Q_{k+1}^* \begin{pmatrix} \chi_{k+2} \\ z(k+3) \end{pmatrix} \\ \begin{pmatrix} w^{(1)}(k) \\ \gamma_{k+2} \end{pmatrix} &= Z_k^* \begin{pmatrix} \gamma_{k+1} \\ w(k+2) \end{pmatrix} \end{aligned} \quad (4.27)$$

with the numbers $z^{(1)}(k+1), w^{(1)}(k)$ and two-dimensional columns χ_{k+3}, γ_{k+2} . Compute

$$f_{k+3}^A = f_{k+3}^V - \chi_{k+3} \gamma_{k+2}^*, \quad \varphi_{k+3}^A = \phi_{k+3}^V. \quad (4.28)$$

– **Recovering of generators**

(a) Set

$$\begin{aligned} g_V^{(1)}(i-2) &= \tilde{g}_V(i), \quad i = 3, \dots, N+2, \\ h_V^{(1)}(j-3) &= \tilde{h}_V(j), \quad j = 4, \dots, N+3, \\ b_V^{(1)}(k-3) &= \tilde{b}_V(k), \quad j = 4, \dots, N+2. \end{aligned}$$

(b) Set

$$\begin{aligned} g_U^{(1)}(i-2) &= \tilde{g}_U(i), \quad i = 3, \dots, N+1, \\ h_U^{(1)}(j-2) &= \tilde{h}_U(j), \quad j = 4, \dots, N+2, \\ b_U^{(1)}(k-2) &= \tilde{b}_U(k), \quad k = 3, \dots, N+1, \\ d_U^{(1)}(k-2) &= \tilde{d}_U(k), \quad k = 3, \dots, N+2. \end{aligned}$$

END

Remark 1 The complete algorithm incorporates the compression technique introduced in [4] to further process the generators returned by the algorithm by computing final upper quasiseparable generators $g_U^{(1)}(i)$ ($i = 1, \dots, N-1$), $h_U^{(1)}(j)$ ($j = 2, \dots, N$), $b_U^{(1)}(k)$ ($k = 2, \dots, N-1$) with orders not greater than one of the matrix U_1 and, moreover, upper triangular generators $g_V^{(1)}(i)$, $h_V^{(1)}(i)$ ($i = 1, \dots, N$), $b_V^{(1)}(k)$ ($k = 1, \dots, N-1$) with orders not greater than two of the matrix V_1 .

In the next section we report the results of numerical experiment to illustrate the performance of the algorithm.

5 Numerical Results

The fast QZ algorithm for eigenvalue computation of structured pencils described in the previous section has been implemented in MATLAB and in Fortran 90.² The program deals with real companion-like pencils by applying the QZ method with single or double shift and it returns as output the list of real or complex conjugate paired approximations of the eigenvalues.

The design of a practical algorithm needs to account for various possible shifting strategies and deflation techniques. Deflation is an important concept in the practical implementation of the QR/QZ iteration. Deflation amounts to setting a small subdiagonal element of the Hessenberg matrix A to zero. This is called deflation because it splits the Hessenberg/triangular matrix pair into two smaller subproblems which may be independently refined further. We say that $a_{k+1,k}$ is negligible if

$$|a_{k+1,k}| \leq \mathbf{u}(|a_{k+1,k+1}| + |a_{k,k}|),$$

and then we set $a_{k+1,k} = 0$ and split the computation into two smaller eigenproblems. Here \mathbf{u} denotes the machine precision. Another kind of deflation can happen in the matrix B and it is related to the occurrence of infinite eigenvalues. If $b_{k,k}$

² Both implementations are available for download at http://www.unilim.fr/pages_perso/paola.boito/software.html.

is numerically zero then there exists at least an infinite eigenvalue and this can be deflated by moving up the zero entry to the top left corner of B . The criterion used in our implementation to check the nullity of $b_{k,k}$ is

$$|b_{k,k}| \leq \mathbf{u} \|B\|.$$

Eligible shift polynomials are generally determined from the (generalized) eigenvalues of the trailing principal submatrices of A and B . We first compute the generalized eigenvalues (α_1, β_1) and (α_2, β_2) of the matrix pair $(A(n-1:n, n-1:n), B(n-1:n, n-1:n))$. If they correspond with a pair of complex conjugate numbers then we set

$$p(z) = (\beta_2 z - \alpha_2)(\beta_2 z - \alpha_2).$$

Otherwise we perform a linear shift, that is, $p(z) = \beta z - \alpha$, where the eigenvalue $\sigma = \alpha/\beta$ is the closest to the value $a_{N,N}/b_{N,n}$.

Our resulting algorithm has been tested on several numerical examples. We begin with some classical polynomials that are meant to test the algorithm for speed and for backward stability. All polynomials are normalized so as to have 2-norm equal to 1. Absolute forward and backward errors for a polynomial $p(x) = \sum_{j=0}^N p_j x^j = p_N \prod_{k=1}^N (x - \alpha_k)$ are defined as

$$\begin{aligned} \text{forward error} &= \max_{k=1,\dots,N} |\alpha_k - \tilde{\alpha}_k|, \\ \text{backward error} &= \max_{j=0,\dots,N} |p_j - \tilde{p}_j|, \end{aligned}$$

where $\{\tilde{\alpha}_k\}_{k=1,\dots,N}$ are the computed roots, and $\{\tilde{p}_j\}_{j=0,\dots,N}$ are the polynomial coefficients reconstructed from the computed roots, working in high precision.

Examples 1 and 2 use the Fortran implementation of Fast QZ, compiled with `gfortran` and running under Linux Ubuntu 14.04 on a laptop equipped with an AMD E-450 APU processor and 3.6 GB memory. All the other tests are based on the MATLAB version of the code and were run on a Mac Book Pro equipped with MATLAB R2016a.

Example 1 Fortran implementation applied to random polynomials. Polynomial coefficients are random real numbers uniformly chosen in $[-1, 1]$. Here N denotes the degree. Table 1 shows forward absolute errors w.r.t. the roots computed by LAPACK, as well as the average number of iterations per eigenvalue and the running times, in seconds, for LAPACK and Fast QZ. All the results are averages over 10 tries for each degree.

In this example, Fast QZ is faster than LAPACK for polynomials of degree larger than 250. (Of course, the results of timing comparisons may vary slightly depending on the machine and architecture). The quadratic growth of the running time for our algorithm is shown in Figure 1.

Example 2 Fortran implementation applied to cyclotomic polynomials. The polynomials used in this example take the form $p(x) = x^N - 1$. Table 2 shows forward absolute errors w.r.t. the roots computed by LAPACK, as well as the average number of iterations per eigenvalue and running times (in seconds). Figure 2 shows a logarithmic plot of the running times for Fast QZ, together with a linear fit.

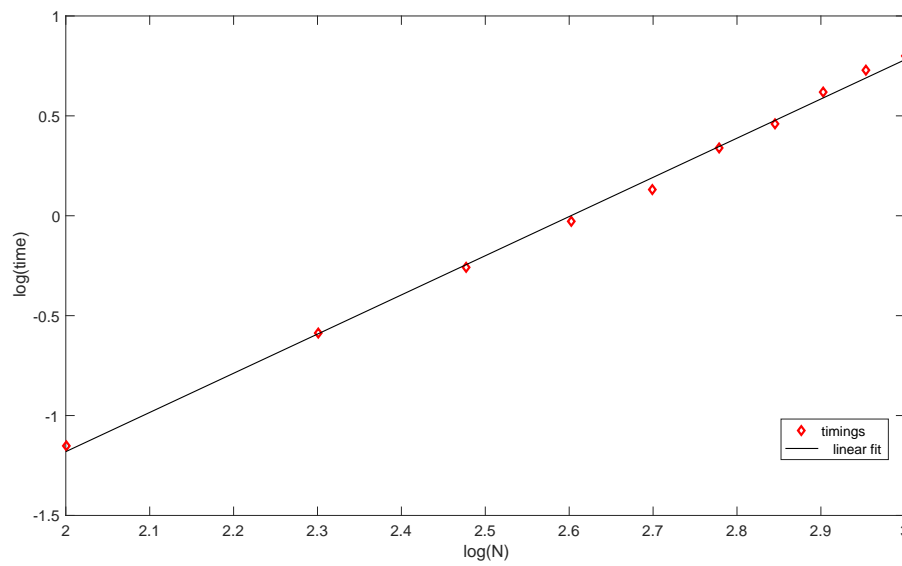


Fig. 1 This is a log-log plot of running times vs. polynomial degree for Example 1. The linear fit has equation $y = 1.96x - 5.10$, which is consistent with the $\mathcal{O}(N^2)$ complexity of Fast QZ.

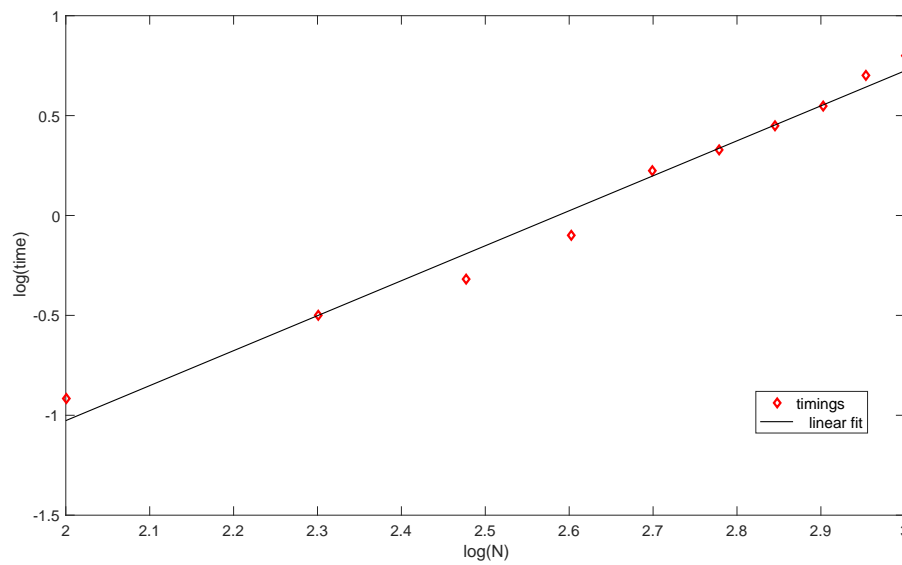


Fig. 2 Log-log plot of running times vs. polynomial degree for Example 2. The linear fit has equation $y = 1.75x - 4.53$.

Table 1 Timings and errors for the Fortran implementation of Fast QZ applied to random polynomials.

N	abs. forward error	average n. it.	Fast QZ time	LAPACK time
50	1.34e-14	1.83	2.15e-2	6.20e-3
100	1.09e-14	1.67	7.08e-2	3.00e-2
200	1.87e-14	1.59	2.60e-1	2.23e-1
250	3.02e-14	1.54	3.94e-1	4.35e-1
300	3.03e-14	1.49	5.55e-1	7.45e-1
400	1.88e-13	1.46	9.33e-1	1.88
500	8.08e-14	1.42	1.36	3.62
600	4.73e-13	1.45	2.18	6.46
700	2.19e-13	1.41	2.88	10.03
800	1.46e-13	1.39	4.16	15.27
900	1.04e-13	1.37	5.35	19.51
1000	1.57e-13	1.40	6.31	28.29

Table 2 Timings and errors for the Fortran implementation of Fast QZ applied to cyclotomic polynomials.

N	abs. forward error	average n. it.	Fast QZ time	LAPACK time
100	3.29e-15	1.38	1.21e-1	2.60e-2
200	8.66e-15	1.25	3.17e-1	1.97e-1
300	1.21e-14	1.19	4.78e-1	5.86e-1
400	1.66e-14	1.16	7.94e-1	1.77
500	2.20e-14	1.15	1.68	4.99
600	2.82e-14	1.12	2.13	5.69
700	3.10e-14	1.12	2.80	10.65
800	3.99e-14	1.10	3.53	13.95
900	3.88e-14	1.10	5.05	19.24
1000	4.72e-14	1.10	6.33	27.52

Example 3 In this example we use a classical set of test polynomials taken from [18]. The polynomials are all of degree 20:

1. the Wilkinson polynomial, i.e., $P(x) = \prod_{k=1}^{20} (x - k)$,
2. the polynomial with roots uniformly spaced in $[-1.9, 1.9]$,
3. $P(x) = \sum_{k=0}^{20} x^k / k!$,
4. the Bernoulli polynomial of degree 20,
5. $P(x) = 1 + x + x^2 + \dots + x^{20}$,
6. the polynomial with roots $2^{-10}, 2^{-9}, \dots, 2^9$,
7. the Chebyshev polynomial of degree 20.

Table 3 shows absolute forward and backward errors for our fast QZ and for classical QZ applied to the companion pencil. For the purpose of computing forward errors we have taken as $\{\alpha_k\}_{k=1, \dots, N}$ either the exact roots, if known, or numerical roots computed with high accuracy.

Forward errors may vary, consistently with the conditioning of the problem. However, backward errors are always of the order of the machine epsilon, which points to a backward stable behavior in practice.

Example 4 The jumping polynomial. This is a polynomial of degree 20 where the coefficients are heavily unbalanced and QZ applied to the companion pencil tends

Table 3 Forward and backward errors for a set of ill-conditioned polynomials. Note that the MATLAB implementation of classical QZ sometimes finds infinite roots, which prevent computation of the backward error. This behavior is denoted by the entry Inf.

$P(x)$	f. err. (fast QZ)	f. err. (classical QZ)	b. err. (fast QZ)	b. err. (classical QZ)
1	28.73	Inf	6.52e-16	Inf
2	5.91e-13	8.07e-13	8.07e-16	1.11e-15
3	5.70	Inf	2.22e-16	Inf
4	3.76e-10	1.83e-12	1.72e-15	1.20e-15
5	3.06e-15	1.09e-15	4.52e-15	1.58e-15
6	1.09e-2	2.30e-3	2.28e-15	3.05e-15
7	5.47e-11	1.68e-11	1.08e-15	1.91e-15

Table 4 Forward and backward errors for several methods applied to a polynomial with highly unbalanced coefficients (Example 4).

method	forward error	backward error
fast QZ	2.78e-15	4.94e-15
classical QZ	1.49e-15	3.22e-15
balanced QR	2.46e-9	5.86e-9
unbalanced QR	1.68e-15	2.72e-15

to work better than computing the eigenvalues of the companion matrix (see also [4]). The polynomial is defined as $p(x) = \sum_{k=0}^{20} p_k x^k$, where $p_k = 10^{6(-1)^{(k+1)}-3}$ for $k = 0, \dots, 20$. Table 4 shows that Fast QZ is just as accurate as classical QZ, and more accurate than the MATLAB command `roots`.

Our algorithm has been tested on several numerical examples resulting from the linearization of nonlinear eigenvalue problems by using Lagrange type interpolation schemes. In particular, if $f: \Omega \subseteq \mathbb{R} \rightarrow \mathbb{R}$ is analytic then increasingly accurate approximations of its zeros can be found by rootfinding methods applied to certain polynomial approximations of f . The unique polynomial of degree less than n interpolating the function $f(z)$ at the N -th roots of unity $z_k = e^{2\pi(k-1)/N}$, $1 \leq k \leq N$, can be expressed as

$$p(z) = (z^N - 1) \sum_{j=1}^n \frac{w_j f_j}{z - z_j},$$

where

$$f_j = f(z_j), \quad w_j = \left(\prod_{k=1, k \neq j}^N (z_j - z_k) \right)^{-1} = z_j / N, \quad 1 \leq j \leq N.$$

In [5] it was shown that the roots of $p(z)$ are the finite eigenvalues of the matrix pencil $F - zG$, $F, G \in \mathbb{C}^{(N+1) \times (N+1)}$, given by

$$F = \begin{bmatrix} 0 & -f_1/\xi_1 & \dots & -f_N/\xi_N \\ w_1 \xi_1 & z_1 & & \\ \vdots & & \ddots & \\ w_N \xi_N & & & z_N \end{bmatrix}, \quad G = \begin{bmatrix} 0 & & & \\ 1 & & & \\ & \ddots & & \\ & & & 1 \end{bmatrix}, \quad (5.29)$$

Table 5 Approximations of the zeros of $f(z) = \sin(z - 0.3) \log(1.2 - z)$. Here N is the number of interpolation points. See Example 5.

N	approx. of 0.2	approx. of 0.3
20	0.2153	0.2841
30	0.2014	0.2986
40	0.20016	0.29983
50	0.200021	0.299978
60	0.2000028	0.2999970
100	0.200000011	0.299999988
200	0.1999999999894	0.30000000000120

where ξ_1, \dots, ξ_N are nonzero real numbers used for balancing purposes. Observe that since the size of the matrices is $N + 1$ we obtain at least two spurious infinite eigenvalues. By a suitable congruence transformation $F \rightarrow F_1 = QFQ^*$ and $G \rightarrow G_1 = QGQ^*$ with Q orthogonal, we generate an equivalent real matrix pair (F_1, G_1) where $G_1 = G$ and F_1 is arrowhead with 2×2 orthogonal diagonal blocks. Then the usual Hessenberg/triangular reduction procedure can be applied by returning a final real matrix pair (\tilde{A}, \tilde{B}) . One infinite eigenvalue can immediately be deflated by simply performing a permutation between the first and the second rows of \tilde{A} and \tilde{B} by returning a final matrix pair (A, B) belonging to the class \mathcal{P}_N . It can be shown that if $\xi_i = \xi$ for all i then this latter matrix pair is the companion pencil associated with the interpolating polynomial expressed in the power basis. Otherwise, if ξ_i are not constant then A is generally a dense Hessenberg matrix which can be represented as a rank one modification of an orthogonal matrix.

In the following examples we test the application of Fast QZ to the barycentric Lagrange interpolation on the roots of unity in order to find zeros of functions or solve eigenvalue problems.

Example 5 This example is discussed in [1]. Consider the function $f(z) = \sin(z - 0.3) \log(1.2 - z)$. We seek the zeros of f in the unit disk; the exact zeros are 0.2 and 0.3. Table 5 shows the computed approximations of these zeros for several values of N (number of interpolation points). The results are consistent with findings in [1], where 50 interpolation points yielded an accuracy of 4 digits.

Example 6 This is also an example from [1]. Define the matrix

$$A = \begin{pmatrix} 3.2 & 1.5 & 0.5 & -0.5 \\ -1.6 & 0.0 & -0.4 & 0.6 \\ -2.1 & -2.2 & 0.2 & -0.1 \\ 20.7 & 9.3 & 3.9 & -3.4 \end{pmatrix}.$$

We want to compute its eigenvalues by approximating the zeros of the polynomial $p(\lambda) = \det(A - \lambda I)$. The exact eigenvalues are 0.2, 0.3, 1.5 and -2 . Interpolation plus Fast QZ using 6 nodes yields all the correct eigenvalues up to machine precision.

One may also apply a similar approach to the computation of the eigenvalues in the unit circle for a larger matrix. See Figures 3 and 4 for tests on two 100×100 matrices with random entries (uniformly chosen in $[-1, 1]$).

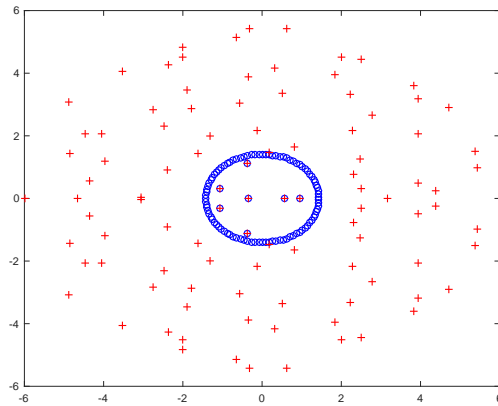


Fig. 3 Eigenvalues of a 100×100 random matrix; see Example 6. The blue circles are the eigenvalues computed via interpolation, the red crosses are the eigenvalues computed by `eig`. Here 120 interpolation nodes were used.

Example 7 We consider some nonlinear eigenvalue problems taken from [2]:

1. `mobile_manipulator`: this 5×5 quadratic matrix polynomial is close to being nonregular;
2. `gen_tpal2`: a real T-palindromic quadratic matrix polynomial of size 16×16 whose eigenvalues lie on the unit circle;
3. `closed_loop`: the eigenvalues of this 2×2 parameterized quadratic polynomial lie inside the unit disc for a suitable choice of the parameter;
4. `relative_pose_5pt`: a 10×10 cubic matrix polynomial which comes from the five point relative pose problem in computer vision. See Figure 5 for a plot of the eigenvalues.

Table 6 shows the distance, in ∞ -norm, between the eigenvalues computed via interpolation followed by Fast QZ and the eigenvalues computed via `polyeig`.

Example 8 Random matrix polynomials: we use matrix polynomials with random coefficients (given by the Matlab function `rand`). Table 7 shows errors with respect to `polyeig` for several values of the degree and of the size of the polynomial.

6 Conclusions

In this paper we have developed and tested a fast structured version of the double-shift QZ eigenvalue method tailored to a particular class of real matrix pencils.

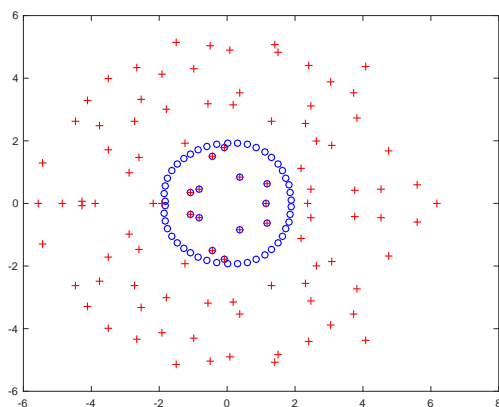


Fig. 4 Eigenvalues of a 100×100 random matrix; see Example 6. The blue circles are the eigenvalues computed via interpolation, the red crosses are the eigenvalues computed by `eig`. Here 60 interpolation nodes were used.

Table 6 Distance between the eigenvalues computed by interpolation+Fast QZ and `polyeig`, for some problems taken from the NLEVP suite (Example 7). Here N is the number of interpolation points. The error for the fourth problem is computed on all the eigenvalues (fourth line) and on the eigenvalues in the unit disk (fifth line, error marked by an asterisk.)

problem	error	N
<code>mobile_manipulator</code>	$2.53e-15$	20
<code>gen_tpal2</code>	$1.61e-9$	50
<code>closed_loop</code>	$1.22e-15$	10
<code>relative_pose_5pt</code>	$2.81e-10$	40
<code>relative_pose_5pt</code>	$8.99e-15^{(*)}$	40

Table 7 Distance between the eigenvalues computed by interpolation+Fast QZ and `polyeig`, for random matrix polynomials of different degrees and sizes (Example 8). The error is computed on the eigenvalues contained in the disk of center 0 and radius 2.

degree	size	error
10	5	$1.12e-11$
10	10	$1.11e-9$
10	20	$2.93e-5$
15	5	$6.98e-9$
15	10	$5.15e-9$
15	20	$3.24e-4$
20	5	$2.13e-10$
20	10	$4.50e-9$

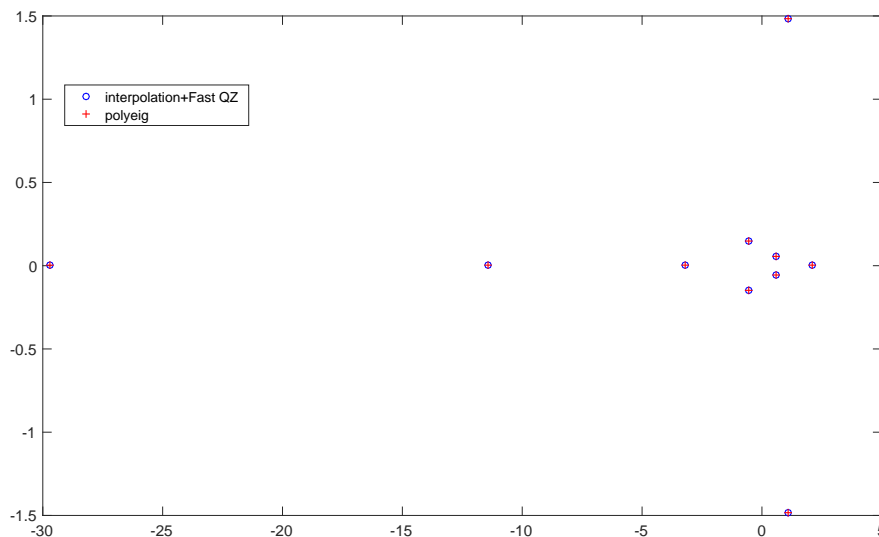


Fig. 5 Eigenvalues of the matrix polynomial `relative_pose_5pt`; see Example 7. The blue circles are the eigenvalues computed via interpolation, the red crosses are the eigenvalues computed by `polyeig`.

This class includes companion pencils, as well as pencils arising from barycentric Lagrange interpolation. Numerical tests confirm the expected complexity gains with respect to the classical method and show that our fast algorithm behaves as backward stable in practice, while retaining an accuracy comparable to the nonstructured method.

We also propose an application to nonlinear eigenvalue problems using interpolation techniques. While preliminary experiments look promising, this approach deserves further investigation, which will be the subject of further work.

Acknowledgements: Thanks to Thomas Mach for useful suggestions concerning the Fortran implementation of Givens transformations.

References

1. A. P. AUSTIN, P. KRAVANJA, AND L. N. TREFETHEN, *Numerical algorithms based on analytic function values at roots of unity*, *SIAM J. Numer. Anal.* 52 (2014), 1795–1821.
2. T. BETCKE, N. J HIGHAM, V. MEHRMANN, C. SCHRÖDER, F. TISSEUR, NLEVP: a collection of nonlinear eigenvalue problems, *ACM Trans. Math. Software*, 39 (2013), 7–28.
3. D. A. BINI, P. BOITO, Y. EIDELMAN, L. GEMIGNANI, AND I. GOHBERG, *A Fast Implicit QR Eigenvalue Algorithm for Companion Matrices*, *Linear Algebra and Applications* 432 (2010), 2006–2031.
4. P. BOITO, Y. EIDELMAN, AND L. GEMIGNANI, Implicit QR for companion-like pencils, *Math. Comp.* 85 (2016), 1753–1774.
5. R. CORLESS, *Generalized companion matrices for the Lagrange basis*, *Proceedings EACA*, 2004.

6. C. DE BOOR, An Empty Exercise, in *ACM SIGNUM Newsletter*, vol. 25 (4), Oct. 1990, 2–6.
7. Y. EIDELMAN, L. GEMIGNANI, AND I. GOHBERG, Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations, *Numerical Algorithms* 47 (2008), 253–273.
8. Y. EIDELMAN AND I. GOHBERG, *On a new class of structured matrices*, *Integral Equations Operator Theory*, 34 (1999), 293–324.
9. Y. EIDELMAN, I. GOHBERG AND I. HAIMOVICI, *Separable type representations of matrices and fast algorithms. Volume 1. Basics. Completion problems. Multiplication and inversion algorithms*, *Operator Theory: Advances and Applications*, Birkhäuser, 2013.
10. Y. EIDELMAN, I. GOHBERG, AND I. HAIMOVICI, *Separable type representations of matrices and fast algorithms. Volume 2. Eigenvalue method*, *Operator Theory: Advances and Applications*, Birkhäuser, 2013.
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.
12. G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.
13. P. LANCASTER, P. PSARRAKOS, *On the pseudospectra of matrix polynomials*, *SIAM J. Matrix Anal. Appl.*, 27 (2005), 115–129.
14. P. W. LAWRENCE, Eigenvalues methods for interpolation bases, *University of West Ontario, Electronic Thesis and Dissertation Repository*, paper 1359 (2013).
15. P. W. LAWRENCE, Fast reduction of generalized companion matrix pairs for barycentric Lagrange interpolants, *SIAM J. Matrix Anal. Appl.* 34 (2013), no. 3, 1277–1300.
16. P. W. LAWRENCE AND R. M. CORLESS, Stability of rootfinding for barycentric Lagrange interpolants, *Numer. Algorithms* 65 (2014), no. 3, 447–464.
17. V. MEHRMANN AND D. WATKINS, *Polynomial eigenvalue problems with Hamiltonian structure*, *Electron. Trans. Numer. Anal.*, 13 (2012), 106–118.
18. K.-C. TOH AND L. N. TREFETHEN, Pseudozeros of polynomials and pseudospectra of companion matrices, *Numer. Math.* 68 (1994), 403–425.
19. D. S. WATKINS, *Fundamentals of matrix computations*, Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], New York, second edition, 2002.

Appendix

In this appendix we give a formal proof of the correctness of the algorithm stated in Section 4. Specifically, we prove the following:

Theorem 1 *Let $(A, B) \in \mathcal{P}_N$ be a matrix pair with an upper Hessenberg matrix $A = V - \mathbf{z}\mathbf{w}^*$ from the class \mathcal{H}_N and an upper triangular matrix $B = U - \mathbf{p}\mathbf{q}^*$ from the class \mathcal{T}_N with the unitary matrices $V \in \mathcal{V}_N, U \in \mathcal{U}_N$ and the vectors $\mathbf{z}, \mathbf{w}, \mathbf{p}, \mathbf{q} \in \mathbb{R}^N$. Let $p(z) = \alpha + \beta z + \gamma z^2 \in \mathbb{R}[z]$ be a polynomial of degree at most 2. Let Q, Z be unitary matrices defined as in (2.8), (2.9) where the matrices Q_i and Z_i , $1 \leq i \leq N - 1$, are generated by the algorithm in Section 4. Then $A_1 = Q^*AZ$ and $B_1 = Q^*BZ$ are upper Hessenberg and upper triangular, respectively, and, moreover, $Q^*p(AB^{-1})\mathbf{e}_1 = \alpha\mathbf{e}_1$ for a suitable scalar $\alpha \in \mathbb{R}$.*

Proof The property $Q^*p(AB^{-1})\mathbf{e}_1 = \alpha\mathbf{e}_1$ easily follows by construction. The proof of the remaining properties is constructive by showing that A_1 is upper Hessenberg and B_1 is upper triangular and then providing structured representations of the entries of their unitary components $V_1 = Q^*VZ$ and $U_1 = Q^*UZ$. We restrict ourselves to consider A_1 and V_1 since the computation of B_1 and U_1 and of the perturbation vectors can be treated in a similar way.

We treat A and V as block matrices with entries of sizes $m_i^A \times n_j^A$, $i, j = 1, \dots, N+3$, where

$$\begin{aligned} m_1^A = \dots = m_N^A = 1, \quad m_{N+1}^A = m_{N+2}^A = m_{N+3}^A = 0, \\ n_1^A = 0, \quad n_2^A = \dots = n_{N+1}^A = 1, \quad n_{N+2}^A = n_{N+3}^A = 0. \end{aligned} \quad (6.30)$$

Relative to this partition the matrix V has diagonal entries

$$\begin{aligned} d_V(1) \text{ to be the } 1 \times 0 \text{ empty matrix,} \\ d_V(k) = \sigma_{k-1}^V = \sigma_{k-1}^A + z(k)w(k-1), \quad k = 2, \dots, N, \\ d_V(N+1), d_V(N+2), d_V(N+3) \text{ to be the } 0 \times 1, 0 \times 0, 0 \times 0 \text{ empty matrices,} \end{aligned} \quad (6.31)$$

upper quasiseparable generators

$$\begin{aligned} \hat{g}_V(k) = g_V(k), \quad k = 1, \dots, N, \quad \hat{g}_V(N+1), g_V(N+2) \text{ to be the } 0 \times 0 \text{ empty matrices,} \\ \hat{h}_V(k) = h_V(k-1), \quad k = 2, \dots, N+1, \\ \hat{h}_V(N+2), \hat{h}_V(N+3) \text{ to be the } 0 \times 0 \text{ empty matrices,} \\ \hat{b}_V(k) = b_V(k-1), \quad k = 2, \dots, N, \\ \hat{b}_V(N+1), \hat{b}_V(N+2) \text{ to be the } r_N^V \times 0, 0 \times 0 \text{ empty matrices} \end{aligned} \quad (6.32)$$

and lower quasiseparable generators

$$\begin{aligned} p_V(k) = z(k), \quad k = 2, \dots, N, \\ p_V(N+1), p_V(N+2) \text{ to be the } 0 \times 1 \text{ empty matrices,} \\ q_V(1) \text{ to be the } 1 \times 0 \text{ empty matrix,} \\ q_V(k) = w(k-1), \quad k = 2, \dots, N+1, \\ a_V(k) = 1, \quad k = 2, \dots, N+1. \end{aligned} \quad (6.33)$$

Relative to the partition (6.30) the matrix A is a block upper triangular matrix with diagonal entries

$$\begin{aligned} d_A(1) \text{ to be the } 1 \times 0 \text{ empty matrix,} \\ d_A(k) = \sigma_{k-1}^A, \quad k = 2, \dots, N, \\ d_A(N+1), d_A(N+2) \text{ to be the } 0 \times 1, 0 \times 0 \text{ empty matrices.} \end{aligned} \quad (6.34)$$

Moreover using (2.1) we obtain upper quasiseparable of the matrix A relative to the partition (6.30) with orders

$$r_k^A = r_k^V + 1, \quad k = 1, \dots, N, \quad r_{N+1}^A = r_{N+2}^A = 0 \quad (6.35)$$

by the formulas

$$\begin{aligned} g_A(k) = [g_V(k) - z(k)], \quad k = 1, \dots, N, \\ g_A(N+1), g_A(N+2) \text{ to be the } 0 \times 0 \text{ empty matrices,} \end{aligned} \quad (6.36)$$

$$\begin{aligned}
h_A(k) &= \begin{bmatrix} h_V(k-1) \\ w(k-1) \end{bmatrix}, \quad k = 2, \dots, N+1, \\
h_A(N+2), h_A(N+3) &\text{ to be the } 0 \times 0 \text{ empty matrix,} \\
b_A(k) &= \begin{pmatrix} b_V(k-1) & 0 \\ 0 & 1 \end{pmatrix}, \quad k = 2, \dots, N,
\end{aligned} \tag{6.37}$$

$b_A(N+1), b_A(N+2)$ to be the $(r_N^V + 1) \times 0, 0 \times 0$ empty matrices.

Using (2.8) and setting

$$\begin{aligned}
\tilde{S}_1^A = \tilde{S}_2^A = I_N, \quad \tilde{S}_i^A = \tilde{Q}_{i-2}^*, \quad i = 3, \dots, N+1, \quad \tilde{S}_{N+2}^A = \tilde{S}_{N+3}^A = I_N, \\
\tilde{T}_1^A = \tilde{T}_2^A = \tilde{T}_3^A = I_N, \quad \tilde{T}_i^A = \tilde{Z}_{i-3}, \quad i = 4, \dots, N+2, \quad \tilde{T}_{N+3}^A = I_N
\end{aligned} \tag{6.38}$$

we get

$$Q^* = \tilde{S}_{N+3}^A \cdots \tilde{S}_1^A, \quad Z = \tilde{T}_1^A \cdots \tilde{T}_{N+3}^A. \tag{6.39}$$

We have

$$\begin{aligned}
\tilde{S}_1^A &= \text{diag}\{S_1^A, I_{N-1}\}, \quad \tilde{S}_2^A = \text{diag}\{S_2^A, I_{N-2}\}; \\
\tilde{S}_k^A &= \text{diag}\{I_{k-2}, S_k^A, I_{N-k}\}, \quad k = 2, \dots, N; \\
\tilde{S}_{N+1}^A &= \text{diag}\{I_{N-2}, S_{N+1}^A\}, \quad \tilde{S}_{N+2}^A = \text{diag}\{I_{N-1}, S_{N+2}^A\}, \quad \tilde{S}_{N+3}^A = \text{diag}\{I_N, S_{N+3}^A\}
\end{aligned}$$

with

$$\begin{aligned}
S_1^A = 1, \quad S_2^A = I_2, \quad S_k^A = Q_{k-2}^*, \quad k = 3, \dots, N+1, \quad S_{N+2}^A = 1, \\
S_{N+3}^A \text{ to be the } 0 \times 0 \text{ empty matrix}
\end{aligned} \tag{6.40}$$

and

$$\begin{aligned}
\tilde{T}_1^A &= \text{diag}\{T_1^A, I_N\}, \quad \tilde{T}_2^A = \text{diag}\{T_2^A, I_{N-1}\}, \quad \tilde{T}_3^A = \text{diag}\{T_2^A, I_{N-2}\}; \\
\tilde{T}_k &= \text{diag}\{I_{k-4}, T_k^A, I_{N-k+1}\}, \quad k = 4, \dots, N+2; \\
\tilde{T}_{N+3}^A &= \text{diag}\{I_{N-1}, T_{N+3}^A\}
\end{aligned}$$

with

$$\begin{aligned}
T_1^A \text{ to be the } 0 \times 0 \text{ empty matrix,} \\
T_2^A = 1, \quad T_3^A = I_2, \quad T_k^A = Z_{k-3}, \quad k = 4, \dots, N+2, \quad T_{N+3}^A = 1.
\end{aligned} \tag{6.41}$$

We treat the lower Hessenberg matrix Q^* as a block matrix with entries of sizes $\tau_i^A \times m_j^A$, $i, j = 1, \dots, N+3$, where

$$\tau_1^A = \tau_2^A = 0, \quad \tau_3^A = \cdots = \tau_{N+2}^A = 1, \quad \tau_{N+3}^A = 0. \tag{6.42}$$

The matrix Q^* has the representation considered in Lemma 31.1 in [10] with the matrices S_k ($k = 1, \dots, N+2$) of sizes $(\tau_1^A + r_1^S) \times m_1^A$, $(\tau_k^A + r_k^S) \times (m_k^A + r_{k-1}^S)$ ($k = 2, \dots, N+1$), $\tau_{N+2}^A \times (m_{N+2}^A + r_{N+1}^S)$, where

$$r_1^S = 1, \quad r_k^S = 2, \quad k = 2, \dots, N, \quad r_{N+1}^S = 1, \quad r_{N+2}^S = 0.$$

We treat the upper Hessenberg matrix Z as a block matrix with entries of sizes $n_i^A \times \nu_j^A$, $i, j = 1, \dots, N+2$, where

$$\nu_1^A = \nu_2^A = \nu_3^A = 0, \quad \nu_4^A = \dots = \nu_{N+3}^A = 1. \quad (6.43)$$

The matrix Z has the representation considered in Lemma 31.1 in [10] with the matrices T_k^A ($k = 1, \dots, N+2$) of sizes $n_1^A \times (\nu_1^A + r_1^*)$, $(n_k^A + r_{k-1}^*) \times (\nu_k^A + r_k^*)$ ($k = 2, \dots, N+1$), $(n_{N+2}^A + r_{N+1}^*) \times \nu_{N+2}^A$, where

$$r_1^* = 0, \quad r_2^* = 1, \quad r_k^* = 2, \quad k = 3, \dots, N+1, \quad r_{N+2}^*.$$

Now we apply the structured multiplication algorithm for quasiseparable representations stated in Corollary 31.2 in [10] in order to determine diagonal entries $\tilde{d}_A(k)$ ($k = 3, \dots, N+2$) and quasiseparable generators $\tilde{q}(j)$ ($j = 3, \dots, N+1$); $\tilde{g}_A(i)$ ($i = 3, \dots, N+2$) of the matrix $A_1 = Q^*AZ$ as well as auxiliary variables $\beta_k^A, f_k^A, \phi_k^A, \varphi_k^A$. The matrix A_1 is obtained as a block one with entries of sizes $\tau_i^A \times \nu_j^A$, $i, j = 1, \dots, N+3$.

For the variables $\beta_k = \beta_k^A, f_k = f_k^A, \phi_k = \phi_k^A$ used in Corollary 31.2 we use the partitions

$$\beta_k^A = [f_k^A \ \phi_k^A], \quad \phi_k^A = [\varphi_k^A \ -\chi_k], \quad k = 2, \dots, N-1, \quad (6.44)$$

with the matrices $f_k^A, \varphi_k^A, \chi_k$ of sizes $2 \times 2, 2 \times r_k^V, 2 \times 1$. For $k = 1, \dots, N-2$ combining Corollary 31.2 with (6.40), (6.41), (6.34) and (6.36),(6.37) we get

$$\begin{aligned} & \begin{pmatrix} \tilde{d}_A(k+3) & \tilde{g}_A(k+3) \\ \tilde{q}(k+3) & \beta_{k+3}^A \end{pmatrix} = \\ Q_{k+1}^* & \begin{pmatrix} f_{k+2}^A & \phi_{k+2}^A h_A(k+2) & \phi_{k+2}^A b_A(k+2) \\ 0 & \sigma_{k+2}^A & g_A(k+3) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+3}^A} \end{pmatrix}, \\ & k = 1, \dots, N-3, \\ & \begin{pmatrix} \tilde{d}_A(N+1) & \tilde{g}_A(N+1) \\ \tilde{q}(N+1) & \beta_{N+1}^A \end{pmatrix} = Q_{N-1}^* (f_N^A \ \phi_N^A h_A(N)) Z_{N-2}. \end{aligned} \quad (6.45)$$

Using (6.44) and (6.37) we get

$$\phi_{k+2}^A h_A(k+2) = \epsilon_{k+2}^A, \quad k = 1, \dots, N-2 \quad (6.46)$$

and

$$\phi_{k+2}^A b_A(k+2) = [\varphi_{k+2}^A b_V(k+2) \ -\chi_{k+2}], \quad k = 1, \dots, N-3 \quad (6.47)$$

with ϵ_{k+2}^A as in (4.14).

Inserting (6.46), (6.47) in (6.45) and using (6.44), (6.36) we obtain

$$\begin{aligned} & \begin{pmatrix} \tilde{d}_A(k+3) & * & * & * \\ \tilde{q}(k+3) & f_{k+3}^A & \varphi_{k+3}^A & -\chi_{k+3} \end{pmatrix} = \\ Q_{k+1}^* & \begin{pmatrix} f_{k+2}^A & \epsilon_{k+2}^A & \varphi_{k+2}^A b_V(k+2) & -\chi_{k+2} \\ 0 & \sigma_{k+2}^A & g_V(k+3) & -z(k+3) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+3}^A} \end{pmatrix}, \\ & k = 1, \dots, N-3. \end{aligned} \quad (6.48)$$

From (6.48) using (4.15) we obtain the relations

$$\begin{pmatrix} \tilde{d}_A(k+3) \\ \tilde{q}(k+3) \end{pmatrix} = Q_{k+1}^* \begin{pmatrix} \Omega_k(1,1) \\ \Omega_k(2,1) \end{pmatrix} \quad (6.49)$$

and

$$Q_{k+1}^* \begin{pmatrix} * & * & * \\ f_{k+3}^A & \varphi_{k+3}^A & -\chi_{k+3} \\ \Omega_k(1:2,2:3) & \varphi_{k+2}^A & -\chi_{k+2} \\ \Omega_k(3,2:3) & g_V(k+3) & -z(k+3) \end{pmatrix} = \quad (6.50)$$

$$Q_{k+1}^* \begin{pmatrix} \Omega_k(1:2,2:3) & \varphi_{k+2}^A & -\chi_{k+2} \\ \Omega_k(3,2:3) & g_V(k+3) & -z(k+3) \end{pmatrix}, \quad k = 1, \dots, N-4.$$

From (6.49) using (4.16) we have

$$\tilde{d}_A(k+3) = (\sigma_k^A)^{(1)}, \quad k = 1, \dots, N-2 \quad (6.51)$$

and

$$\tilde{q}(k+3) = 0, \quad k = 1, \dots, N-2. \quad (6.52)$$

The formulas (6.42) and (6.43) mean that $(\sigma_k^A)(1)$, $k = 1, \dots, N-2$ are subdiagonal entries of the matrix A_1 (treated as an usual scalar matrix). The equalities (6.52) imply that A_1 is an upper Hessenberg matrix.

Next we apply the structured multiplication algorithm stated in Lemma 31.1 in [10] to compute (block) upper quasiseparable generators $\tilde{g}_V(i)$ ($i = 1, \dots, N+2$), $\tilde{h}_V(j)$ ($j = 2, \dots, N+3$), $\tilde{b}_V(k)$ ($k = 2, \dots, N+2$) with orders

$$\tilde{r}_1^V = r_1^V, \tilde{r}_2^V = r_2^V + 1, \quad \tilde{r}_k^V = r_k^V + 2, \quad k = 3, \dots, N, \quad \tilde{r}_{N+1}^V = 2, \tilde{r}_{N+2}^V = 1$$

and diagonal entries $\tilde{d}_{V_1}(k)$ ($k = 1, \dots, N+3$) of the matrix $V_1 = Q^*VZ$. The matrix V_1 is obtained as a block one with entries of sizes $\tau_i^A \times \nu_j^A$, $i, j = 1, \dots, N+3$, where the numbers τ_i^A, ν_j^A are defined in (6.42), (6.43).

Using Lemma 31.1 and (6.40), (6.41) we obtain that

$$\begin{pmatrix} \tilde{d}_V(k) & \tilde{g}_V(k) \\ * & \Gamma_k \end{pmatrix} = \quad (6.53)$$

$$\begin{pmatrix} Q_{k-2}^* & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} f_{k-1}^V & \phi_{k-1}^V h_V(k-1) & \phi_{k-1}^V b_V(k-1) \\ z(k)\alpha_{k-1} & \sigma_{k-1}^V & g_V(k) \\ \alpha_{k-1} & w^*(k-1) & 0 \end{pmatrix} \begin{pmatrix} Z_{k-2} & 0 \\ 0 & I_{r_k^V} \end{pmatrix},$$

$$\Gamma_k = \begin{bmatrix} f_k^V & \phi_k^V \\ \alpha_k & 0 \end{bmatrix}, \quad k = 4, \dots, N,$$

together with the relation (4.26).

From (6.53) we find that the auxiliary matrices α_k ($k = 3, \dots, N$) satisfy the relations

$$\alpha_3 = (w(1) w(2)), \quad (* \alpha_k) = (\alpha_{k-1} w(k-1)) Z_{k-3}, \quad k = 4, \dots, N.$$

Comparing this with (4.4), (4.21) we get

$$\alpha_k = \gamma_{k-1}^*, \quad k = 3, \dots, N. \quad (6.54)$$

Thus using (6.54) and (6.53) we obtain (4.24), (4.25).

Next we show that the auxiliary variables f_k^A, φ_k^A ($k = 3, \dots, N$) may be determined via relations (4.5), (4.28). Take γ_2 as in (4.4) and assume that for some k with $1 \leq k \leq N - 2$ the relations

$$f_{k+2}^A = f_{k+2}^V - \chi_{k+2}\gamma_{k+1}^*, \quad \varphi_{k+2}^A = \phi_{k+2}^V \quad (6.55)$$

hold. By (4.15) and (6.50) we have

$$\begin{aligned} & \begin{pmatrix} \tilde{d}_A(k+3) & * & * \\ 0 & f_{k+3}^A & \varphi_{k+3}^A \end{pmatrix} = \\ Q_{k+1}^* & \begin{pmatrix} f_{k+2}^A & \epsilon_{k+2}^A & \phi_{k+2}^V b_V(k+2) \\ 0 & \sigma_{k+2}^A & g_V(k+3) \end{pmatrix} \begin{pmatrix} Z_k & 0 \\ 0 & I_{r_{k+3}^V} \end{pmatrix}. \end{aligned} \quad (6.56)$$

Using (4.23) and (4.14) we get

$$\begin{aligned} & \begin{pmatrix} f_{k+2}^A & \epsilon_{k+2}^A \\ 0 & \sigma_{k+2}^A \end{pmatrix} = \\ & \begin{pmatrix} f_{k+2}^V - \chi_{k+2}\gamma_{k+1}^* & \phi_{k+2}^V h_V(k+2) - \chi_{k+2}w(k+2) \\ z(k+3)\gamma_{k+1}^* - z(k+3)\gamma_{k+1}^* & \sigma_{k+2}^V - z(k+3)w(k+2) \end{pmatrix}. \end{aligned} \quad (6.57)$$

Thus combining (6.56) and (6.57) together and using (4.24), (4.25) and (4.27) we obtain (4.28).