A CMV–BASED EIGENSOLVER FOR COMPANION MATRICES

R. BEVILACQUA*, G. M. DEL CORSO*, AND L. GEMIGNANI*¶

Abstract.
In this paper we present a novel matrix method for polynomial rootfinding. The roots are approximated by computing the eigenvalues of a permuted version of the companion matrix associated with the polynomial in block upper Hessenberg form with possibly nonsquare subdiagonal blocks. It is shown that this form, referred to as a lower staircase form of the companion matrix in reference to its characteristic appearance, is well suited for the application of the QR eigenvalue algorithm. In particular, each matrix generated under this iteration is block upper Hessenberg and, moreover, all its submatrices located in a specified upper triangular portion are of rank two at most with entries represented by means of four given vectors. By exploiting these properties we design a fast and computationally simple structured QR iteration which computes the eigenvalues of a companion matrix of size n in lower staircase form using $O(n^2)$ flops and $O(n)$ memory storage. This iteration is theoretically faster than other fast variants of the QR iteration for companion matrices in customary Hessenberg form. Numerical experiments show the efficiency and the accuracy of the proposed approach.
AMS classification: 65F15

Key words. Companion matrix, QR eigenvalue algorithm, CMV matrix, rank structure, complexity, accuracy.

1. Introduction.

Cleve Moler raised the question about the efficiency of the MatLab* function `roots` for approximating the roots of an algebraic equation [30, 31]. Univariate polynomial rootfinding is a fundamental and classic mathematical problem. A wide bibliography, some history, applications and algorithms can be found in [29, 32]. `Roots` approximates the zeros of a polynomial by computing the eigenvalues of the associated companion matrix, which is a unitary plus a rank–one matrix in upper Hessenberg form constructed from the coefficients of the polynomial. The analysis and the design of efficient eigensolvers for companion matrices have substantially influenced the recent development of numerical methods for matrices with rank structure [39, 21]. Roughly speaking, a matrix $A \in \mathbb{C}^{n\times n}$ is rank structured if all its off–diagonal submatrices are of small rank.

This paper stems from two research lines aimed at the effective solution of certain eigenproblems for companion–like matrices arising in polynomial rootfinding. The first one begins with the exploitation of the structure of companion-like matrices under the QR eigenvalue algorithm. In the recent years many authors have argued that the rank structure of a matrix A in upper Hessenberg form propagates along the QR iteration whenever A can be expressed as a low rank correction of a unitary or Hermitian matrix. However, despite the common framework, there are several significant differences between the Hermitian and the unitary case so that for the latter suitable techniques are required in order to retain the unitary property of the unperturbed component. The second line originates from the treatment of the unitary eigenproblem. It has been observed in the seminal paper [13] that a five–diagonal banded form of a unitary matrix can be obtained from a suitable rearrangement of the Schur parametrization of its Hessenberg reduction. Moreover, this band reduction rather than the Hessenberg form itself leads to a QR–type algorithm which is close

*Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127 Pisa, Italy, {bevilacq,delcorso,l.gemignani}@di.unipi.it
¶This work was partially supported by GNCS-INDAM.
*Matlab is a registered trademark of The MathWorks, Inc..

to the Hermitian tridiagonal QR algorithm as it maintains the band shape of the initial matrix at all steps. The five–diagonal form exhibits a staircase shape and it is generally referred to as the CMV form of a unitary matrix since the paper [14] by Cantero, Moral and Velázquez enlightens the connection between these banded unitary matrices and certain sequences of Szegö polynomials orthogonal on the unit circle (compare also with [28]). The present work lies at the intersection of these two strands and specifically aims at incorporating the CMV technology for the unitary eigenproblem in the design of fast QR–based eigensolvers for companion matrices.

The first fast structured variant of the QR iteration for companion matrices was proposed in [6]. The invariance of the rank properties of the matrices generated by the QR scheme is captured by means of six vectors which specify the strictly upper triangular part of these matrices. The vectors are partly determined from a structured representation of the inverse of the iteration matrix which has Hessenberg form. The representation breaks down for reducible Hessenberg matrices and, hence, the price paid to keep the algorithm simple is a progressive deterioration in the limit of the accuracy of computed eigenvalues. Overcoming this drawback is the main subject of many subsequent papers [5, 7, 8, 16, 36], where more refined parametrizations of the rank structure are employed. While this leads to numerically stable methods, it also opens the way to involved algorithms which do not improve the timing performance for small to moderate size problems and/or are sometime difficult to generalize to the block matrix/pencil case. Actually the comparison of running times versus polynomial degrees for Lapack and structured QR implementations shows crossover points for moderately large problems with degrees located in the range between $n = 100$ and $n = 200$. Further, the work [24] is so far the only generalization to focus on block companion matrices, but the proposed method is inefficient wrt block size.

The comparison is astonishingly unpleasant if we account for the simplicity and the effectiveness of some adaptations of the QR scheme for perturbed Hermitian matrices [20, 38]. In order to alleviate this difference, approaches based on LR-type algorithms have been recently proposed. Zhlobich [40] develops a variant of the differential qd algorithm for certain rank structured matrices which shows promising features when applied to companion matrices. A proposal for improving the efficiency of structured matrix–based rootfinders is presented in [3] where a LU–type eigenvalue algorithm for some Fiedler companion matrices is devised. In particular, the numerical results shown in [3] indicate that their nonunitary method is at least four times faster than the unitary variant presented in [5]. The application of the differential qd algorithm to the five–diagonal Fiedler companion matrix [31] is also discussed by Parlett in his abstract at the last Householder Conference [33]. However, although Fiedler companion matrices are potentially suited for the design of accurate polynomial rootfinders [17], it is clear that LU–type methods can suffer from numerical instabilities.

Our contribution is to show that a comparable speedup can be achieved in the framework of QR algorithms by using an alternative entrywise and data–sparse representation of companion matrices. Motivated by the treatment of the unitary eigenproblem the approach pursued here moves away from the standard scheme where a nonsymmetric matrix is converted in Hessenberg form before computing its eigenvalues by means of the QR algorithm. On the contrary, here we focus on an alternative preliminary reduction of a companion matrix $A \in \mathbb{C}^{n\times n}$ into a different lower staircase form, that is, a block upper Hessenberg form with 1-by-1 or 2-by-2 diagonal blocks and possibly nonsquare subdiagonal blocks with one nonzero column at most. More specifically, recall that a companion matrix $A \in \mathbb{C}^{n\times n}$ can be expressed as a rank–one correction of a unitary matrix U generating the circulant matrix algebra. The transformation of U by unitary similarity into its five–diagonal CMV representation induces a corresponding reduction of the matrix A into the desired lower staircase form.

This form encompassed in the block upper Hessenberg partition of the matrix is invariant under the QR eigenvalue algorithm [2]. Moreover, the reduction of the unitary component in CMV form yields additional properties of the sequence of the matrices $\{A_k\}$, $A_0 = A$, generated by the iterative process. It is shown that each matrix A_k admits a QR factorization $A_k = Q_k R_k$ where the unitary factor Q_k has a five–diagonal CMV form. From this, mostly because of the banded structure of Q_k, it follows that each A_k inherits a simplified rank structure. All the submatrices of A_k located in a given upper triangular portion of the matrix are of rank two at most and the entries can be expressed by using two rank–one matrices. This yields a data sparse representation of each matrix stored as a banded staircase matrix plus the upper triangular portion specified by four vectors. The decomposition is well suited to capture the structural properties of the matrix and yet it is very easy to manipulate and update for computations.

In this paper we shall develop a fast adaptation of the QR eigenvalue algorithm for companion matrices that exploits this representation by requiring $O(n)$ arithmetic operations and $O(n)$ memory storage per step. The novel unitary variant has a cost of about $80n + O(1)$ flops per iteration, and hence it is theoretically twice as fast as the algorithm in [5] (compare with the cost analysis shown in [21] where a cost of $189n + O(1)$ is reported without counting the time spent for additional operations such as factorizing small matrices or computing the unitary matrices involved in the QR step). Here for the sake of comparison with [5] flop stands for an axpy operation like $e = c + a * b$. The main complexity of our algorithm lies in updating the banded staircase component of each matrix. Since the width of the band is small compared with the order of the matrix, the amount of computation time spent on the updating of the banded matrix, in each step, is quite modest. Moreover since the representation is entrywise the deflation process can be implemented efficiently simply comparing the entries with the corresponding diagonal entries scaled by a suitable tolerance. Thus, the speedup measured experimentally is even better than theoretical estimates and actually our algorithm is about four times faster than the variant in [5], while achieving a comparable accuracy.

The paper is organized as follows. In Section 2, we first introduce the main problem and then we briefly set up the preliminaries, basic reductions and notation that we will use throughout the rest of the paper. The structural properties of of the matrices generated by the QR iteration applied to the considered permuted form of a companion matrix are analyzed in Section 3. In Section 4 we present our fast adaptation of the shifted QR algorithm for companion matrices and report the results of numerical experiments. Finally, in Section 5 the conclusion and further developments are drawn.

2. Problem Statement and Preliminaries. We study the problem of approximating the zeros of a univariate polynomial $p(z)$ of degree n,

$$p(z) = p_0 + p_1 z + \ldots + p_n z^n, \quad (p_n \neq 0).$$

Polynomial rootfinding via eigensolving for an associated companion matrix is an increasingly popular approach. From the given $n-$th degree polynomial $p(z)$ we can set up the associated companion matrix $C \in \mathbb{C}^{n\times n}$ in upper Hessenberg form,

$$C = C(p) = \begin{bmatrix} -\frac{p_{n-1}}{p_n} & -\frac{p_{n-2}}{p_n} & \dots & -\frac{p_0}{p_n} \\ 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \vdots \\ & & 1 & 0 \end{bmatrix}.$$

Since

$$p_n \det(zI - C) = p(z),$$

we obtain approximations of the zeros of $p(z)$ by applying a method for eigenvalue approximation to the associated companion matrix C. The (single) shifted QR algorithm

$$\begin{cases} A_s - \rho_s I_n = Q_s R_s \\ A_{s+1} = Q_s^H A_s Q_s, \quad s \geq 0, \end{cases} \tag{2.1}$$

is the standard algorithm for computing the eigenvalues of a general matrix $A = A_0 \in \mathbb{C}^{n\times n}$ [25] and can be applied to compute the zeros of $p(z)$ setting $A_0 := C$. This is basically the approach taken by the MatLab function `roots`, which also incorporates matrix balancing preprocessing and the use of double shifted variants of (2.1) for real polynomials.

The QR method is not readily amenable to exploit the structure of the companion matrix. In the recent years many fast adaptations of the QR iteration (2.1) applied to an initial companion matrix $A_0 = C$ have been proposed that are based on the decomposition of C as a rank–one correction of a unitary matrix, that is,

$$C = U - \boldsymbol{e}_1 \boldsymbol{p}^H = \begin{bmatrix} 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \vdots \\ & & 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \left[\frac{p_{n-1}}{p_n}, \frac{p_{n-2}}{p_n}, \dots, \frac{p_0}{p_n} + 1 \right].$$

In this paper we further elaborate on this decomposition by developing a different representation. The so–called Schur parametrization of a unitary upper Hessenberg matrix with positive subdiagonal entries [26] yields a representation of U as product of Givens rotations. For a given pair $(\gamma, k) \in \mathbb{D} \times \mathbb{I}_{n-1}$, $\mathbb{D} = \{z \in \mathbb{C}\colon |z| < 1\}$, $\mathbb{I}_n = \{1, 2, \dots, n\}$, we set

$$\mathcal{G}_k(\gamma) = I_{k-1} \oplus \begin{bmatrix} \bar{\gamma} & \sigma \\ \sigma & -\gamma \end{bmatrix} \oplus I_{n-k-1} \in \mathbb{C}^{n\times n}, \tag{2.2}$$

where $\sigma \in \mathbb{R}, \sigma > 0$ and $|\gamma|^2 + \sigma^2 = 1$. Similarly, if $\gamma \in \mathbb{S}^1 = \{z \in \mathbb{C}\colon |z| = 1\}$ then denote

$$\mathcal{G}_n(\gamma) = I_{n-1} \oplus \gamma \in \mathbb{C}^{n\times n}.$$

Observe that $\mathcal{G}_k(\gamma)$, $1 \leq k \leq n$, is a unitary matrix. Furthermore, it can be easily seen that

$$U = \mathcal{G}_1(0) \cdot \mathcal{G}_2(0) \cdots \mathcal{G}_{n-1}(0) \cdot \mathcal{G}_n(1)$$

gives the unique Schur parametrization of U.

A suitable rearrangement of this parametrization is found by considering the permutation defined by

$$\pi \; : \; \mathbb{I}_n \to \mathbb{I}_n, \quad \pi(1) = 1; \; \pi(j) = \begin{cases} k+1, & \text{if } j = 2k; \\ \\ n-k+1, & \text{if } j = 2k+1. \end{cases}$$

Let $P \in \mathbb{R}^{n \times n}$, $P = (\delta_{i,\pi(j)})$ be the permutation matrix associated with π, where δ denotes the Kronecker delta. The following observation provides the starting point of our approach.

LEMMA 2.1. *The $n \times n$ unitary matrix $\widehat{U} = P^T \cdot U \cdot P = (\widehat{u}_{i,j})$ satisfies*

$$\widehat{u}_{i,j} = \begin{cases} 1 \iff (i,j) \in \mathcal{J}_n \cup (2,1); \\ 0 \text{ elsewhere}, \end{cases}$$

where for n even and odd, respectively, we set

$$\mathcal{J}_n = \{(2k, 2k-2), 2 \le k \le n/2\} \cup \{(2k-1, 2k+1), 1 \le k \le n/2 - 1\} \cup \{(n-1, n)\},$$

and

$$\mathcal{J}_n = \{(2k, 2k-2), 2 \le k \le (n-1)/2\} \cup \{(2k-1, 2k+1), 1 \le k \le (n-1)/2\} \cup \{(n, n-1)\}.$$

Moreover, it holds

$$\widehat{U} = \mathcal{G}_1(0) \cdot \mathcal{G}_3(0) \cdots \mathcal{G}_{2\lfloor \frac{n+1}{2} \rfloor - 1}(\delta_{1,\mathrm{mod}(n,2)}) \cdot \mathcal{G}_2(0) \cdot \mathcal{G}_4(0) \cdots \mathcal{G}_{2\lfloor \frac{n}{2} \rfloor}(1 - \delta_{1,\mathrm{mod}(n,2)}).$$

Proof. The first characterization of $\widehat{U}$ is a direct calculation from

$$\widehat{u}_{i,j} = u_{\pi(i),\pi(j)}, \quad 1 \le i, j \le n.$$

The factorized decomposition is found by computing the QR factorization of the matrix $\widehat{U}$ by using Givens rotations. □

The transformation $U \to \widehat{U}$ induces the reduction of the companion matrix C into a different form $\widehat{C}$ defined by

$$\widehat{C} = P^T \cdot C \cdot P = P^T U P - P^T \boldsymbol{e}_1 \boldsymbol{p}^H P = \widehat{U} - \boldsymbol{e}_1 \widehat{\boldsymbol{p}}^H. \tag{2.3}$$

We shall emphasize the importance of this reduction by showing that the use of the QR scheme applied to $\widehat{C}$ instead of C for the approximation of the zeros of $p(z)$ has several advantages. These are due to some structural properties/shapes of both $\widehat{U}$ and $\widehat{C}$ that propagate and/or play a role along the QR iteration. In the next subsections we give a look at these features.

2.1. Unitary Matrices in CMV Form. The following definition identifies an important class of structured unitary matrices.

DEFINITION 2.2. *[14] For a given coefficient sequence $(\gamma_1, \ldots, \gamma_{n-1}, \gamma_n) \in \mathbb{D}^{n-1} \times \mathbb{S}^1$ we introduce the unitary block diagonal matrices*

$$\mathcal{L} = \mathcal{G}_1(\gamma_1) \cdot \mathcal{G}_3(\gamma_3) \cdots \mathcal{G}_{2\lfloor \frac{n+1}{2} \rfloor - 1}(\gamma_{2\lfloor \frac{n+1}{2} \rfloor - 1}), \quad \mathcal{M} = \mathcal{G}_2(\gamma_2) \cdot \mathcal{G}_4(\gamma_4) \cdots \mathcal{G}_{2\lfloor \frac{n}{2} \rfloor}(\gamma_{2\lfloor \frac{n}{2} \rfloor}),$$

and define

$$\mathcal{C} = \mathcal{L} \cdot \mathcal{M} \tag{2.4}$$

as the CMV matrix associated with the prescribed coefficient list.

The decomposition (2.4) of a unitary matrix was first investigated for eigenvalue computation in [13]. The shape of CMV matrices is analyzed in [28] where the next definition is given.

DEFINITION 2.3. *[28] A matrix $A \in \mathbb{C}^{n\times n}$ has CMV shape if the possibly nonzero entries exhibit the following pattern where $+$ denotes a positive entry:*

$$A = \begin{bmatrix} \star & \star & + & & & & & \\ + & \star & \star & & & & & \\ & \star & \star & \star & + & & & \\ & + & \star & \star & \star & & & \\ & & & \star & \star & \star & + & \\ & & & + & \star & \star & \star & \\ & & & & & \star & \star & \star \\ & & & & & + & \star & \star \end{bmatrix}, \qquad (n = 2k),$$

or

$$A = \begin{bmatrix} \star & \star & + & & & & \\ + & \star & \star & & & & \\ & \star & \star & \star & + & & \\ & + & \star & \star & \star & & \\ & & & \star & \star & \star & + \\ & & & + & \star & \star & \star \\ & & & & & \star & \star \end{bmatrix}, \qquad (n = 2k-1).$$

The definition is useful for computational purposes since shapes are easier to check than comparing the entries of the matrix. Obviously, CMV matrices have a CMV shape and, conversely, it is shown that a unitary matrix with CMV shape is CMV [15]. From Lemma 2.1 it follows that $\widehat{U}$ is a CMV matrix and, therefore, it has a CMV shape. For instance in the case $n = 8$ the nonzero pattern of $\widehat{U}$ looks as follows:

$$\widehat{U} = \left[\begin{array}{cc|cc|cc|cc} & & 1 & & & & & \\ 1 & & & & & & & \\ \hline & & & & 1 & & & \\ & 1 & & & & & & \\ \hline & & & & & & 1 & \\ & & & 1 & & & & \\ \hline & & & & & & & 1 \\ & & & & & 1 & & \end{array}\right].$$

The positiveness of the complementary parameters σ_k in (2.2) as well as of the entries marked with $+$ in Definition 2.3 is necessary to establish the connection of CMV matrices with corresponding sequences of orthogonal polynomials on the unit circle [28]. From the point of view of eigenvalue computation, however, this condition can be relaxed. In [4] we simplify the above definition by skipping the positiveness condition on the entries denoted as $+$. The fairly more general class of matrices considered in [4] is referred to as CMV–like shaped matrices. It is shown that the block Lanczos method can be used to reduce a unitary matrix into the direct sum of CMV–like shaped matrices. Furthermore, some rank properties of unitary CMV–like shaped matrices can be put in evidence by invoking the following classical *nullity theorem* [22].

THEOREM 2.4. *Suppose $A \in \mathbb{C}^{n\times n}$ is a nonsingular matrix and let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be nonempty proper subsets of $\mathbb{I}_n := \{1,\dots,n\}$. Then*

$$\mathrm{rank}(A^{-1}(\boldsymbol{\alpha},\boldsymbol{\beta})) = \mathrm{rank}(A(\mathbb{I}_n \setminus \boldsymbol{\beta}; \mathbb{I}_n \setminus \boldsymbol{\alpha})) + |\boldsymbol{\alpha}| + |\boldsymbol{\beta}| - n,$$

where, as usual, $|J|$ denotes the cardinality of the set J, and $A^{-1}(\alpha,\beta)$ denotes the minor of A^{-1} obtained taking the rows and columns in α and β respectively.

The next property of unitary CMV–like shaped follows as a direct consequence.

COROLLARY 2.5. *Let $A \in \mathbb{C}^{n\times n}$ be a unitary CMV–like shaped matrix. Then we have*

$$\mathrm{rank}(A(2j+1:2(j+1), 2j:2j+1)) = 1, \quad 1 \le j \le \lfloor \frac{n}{2} \rfloor - 1.$$

Proof. From Theorem 2.4 we obtain that

$$0 = \mathrm{rank}(A(1:2j, 2(j+1):n)) = \mathrm{rank}(A^H(2(j+1):n, 1:2j)) =$$
$$\mathrm{rank}(A(2j+1:n, 1:2j+1) + (n-1) - n = \mathrm{rank}(A(2j+1:n, 1:2j+1) - 1.$$

□

In passing, it is worth noting that this property is also useful to show that CMV–like shaped matrices admit an analogous factorization (2.4) in terms of generalized Givens rotations of the form

$$\mathcal{R}_k(\gamma,\sigma) = I_{k-1} \oplus \begin{bmatrix} \bar{\gamma} & \sigma \\ \bar{\sigma} & -\gamma \end{bmatrix} \oplus I_{n-k-1} \in \mathbb{C}^{n\times n}, \quad 1 \le k \le n-1, \tag{2.5}$$

where $\gamma, \sigma \in \mathbb{D} \cup \mathbb{S}^1$ and $|\gamma|^2 + |\sigma|^2 = 1$. When σ is a real and positive number, $\mathcal{R}_k(\gamma,\sigma) = \mathcal{G}_k(\gamma)$.

2.2. Staircase Matrices. CMV–like shaped matrices can be partitioned in a block upper Hessenberg form with 1–by–1 or 2–by–2 diagonal blocks. The additional zero structure of the subdiagonal blocks yields the given staircase shape.

DEFINITION 2.6. *[2] The matrix $A = (a_{i,j}) \in \mathbb{C}^{n\times n}$ is said to be staircase if $m_j(A) \ge m_{j-1}(A)$, $2 \le j \le n$, where*

$$m_j(A) = \max\{j, \max_{i>j}\{i\colon a_{i,j} \neq 0\}\}.$$

The matrix A is said to be full staircase if there are no zero elements within the stair.

Staircase linear systems are ubiquitous in applications [23]. Staircase matrix patterns can also be exploited for eigenvalue computation [2]. In order to account for the possible zeroing in the strictly lower triangular part of the matrix modified under the QR iteration we introduce the following definition.

DEFINITION 2.7. *The lower staircase envelope of a matrix $A = (a_{i,j}) \in \mathbb{C}^{n\times n}$ is the sequence $(\tilde{m}_1(A),\dots,\tilde{m}_n(A))$, where $\tilde{m}_1(A) = m_1(A)$ and*

$$\tilde{m}_j(A) = \max\{j, \tilde{m}_{j-1}(A), \max_{i>j}\{i\colon a_{i,j} \neq 0\}\}, \quad 2 \le j \le n.$$

From Definition 2.3 it is found that for a CMV matrix A we have

$$\tau_1 \colon = \tilde{m}_1(A) = 2 \tag{2.6}$$

and, moreover, for $1 \le k \le \lfloor \frac{n+1}{2} \rfloor - 1$,

$$\tau_{2k} \colon = \tilde{m}_{2k}(A) = \min\{2(k+1), n\}, \ \tau_{2k+1} \colon = \tilde{m}_{2k+1}(A) = \min\{2(k+1), n\}. \tag{2.7}$$

The same relations hold for the perturbed companion matrix $\widehat{C}$ given in (2.3). Similarly, a CMV–like shaped matrix A satisfies

$$\tilde{m}_j(A) \le \tau_j, \quad 1 \le j \le n.$$

The lower staircase envelope of a matrix $A = A_0$ form is preserved under the QR iteration (2.1) in the sense that [2]

$$\tilde{m}_j(A_{s+1}) \le \tilde{m}_j(A_s), \quad 1 \le j \le n, s \ge 0. \tag{2.8}$$

In particular, if $A_0 = \widehat{C}$ we deduce that

$$\tilde{m}_j(A_s) \le \tau_j, \quad 1 \le j \le n, s \ge 0. \tag{2.9}$$

Remark 2.8. *A simple proof of* (2.8) *follows by assuming that the matrix $A_s - \sigma_s I_n$ in* (2.1) *and, hence, a fortiori R_s is invertible. Clearly, this might not always be the case but, however, it is well known that the one–parameter matrix function $A_s - \lambda I_n$ is analytic in λ and an analytic QR decomposition $A_s - \lambda I_n = Q_s(\lambda)R_s(\lambda)$ of this analytic matrix function exists [19]. For any given fixed initial pair $(Q_s(\sigma_s), R_s(\sigma_s))$ we can find a branch of the analytic QR decomposition of $A_s - \lambda I_n$ that passes through $(Q_s(\sigma_s), R_s(\sigma_s))$. Following this path makes it possible to extend the proof of the properties that are closed in the limit. This is for instance the case for the rank properties and zero patterns of submatrices located in the lower triangular corner.*

Remark 2.9. *It is worth pointing out that according to the definition stated in [2] $\widehat{C}$ is not* full staircase *as there are many zero entries within the stair. In this case there is a fill-in at the first steps of the QR algorithm and, after a number of iterations the QR iterates will be full staircase matrices, the staircase being the lower staircase envelope of $\widehat{C}$.*

For Hermitian and unitary matrices the lower staircase envelope also determines a zero pattern or a rank structure in the upper triangular part. Relation (2.8) implies the invariance of this pattern/structure by the QR algorithm. A formal proof is given in [2] for Hermitian matrices and in [13] for unitary CMV–like shaped matrices. In the next section we generalize these properties to the permuted companion matrix (2.3). This allows the efficient implementation of the QR iteration (2.1) for the computation of the zeros of $p(z)$.

3. Structural Properties under the QR Iteration. In this section we perform a thorough analysis of the structural properties of the matrices A_s, $s \ge 0$, generated by the QR iteration (2.1) applied to the permuted companion matrix $A_0 = \widehat{C}$ defined in (2.3). It is shown that putting together the staircase shape of $\widehat{C}$ with the CMV form of the unitary component $\widehat{U}$ imposes strong requirements about the structure of the matrices A_s which enable a representation of the matrix entries using a linear number of parameters.

Since from (2.3) we have

$$A_0 = \widehat{C} = \widehat{U} - \boldsymbol{e}_1 \,\widehat{\boldsymbol{p}}^H := U_0 - \boldsymbol{z}_0 \boldsymbol{w}_0^H, \tag{3.1}$$

then by applying the QR algorithm (2.1) we find that

$$A_{s+1} = Q_s^H A_s Q_s = Q_s^H (U_s - \boldsymbol{z}_s \boldsymbol{w}_s^H) Q_s = U_{s+1} - \boldsymbol{z}_{s+1} \boldsymbol{w}_{s+1}^H, \quad s \geq 0, \tag{3.2}$$

where

$$U_{s+1} :\, = Q_s^H U_s Q_s, \quad \boldsymbol{z}_{s+1} :\, = Q_s^H \boldsymbol{z}_s, \quad \boldsymbol{w}_{s+1} :\, = Q_s^H \boldsymbol{w}_s. \tag{3.3}$$

The shifting technique is aimed at speeding up the reduction of the matrix A_0 into a block upper triangular form. A matrix $A \in \mathbb{C}^{n \times n}$ is *reduced* if there exists an integer k, $1 \leq k \leq n-1$, such that

$$A = \begin{bmatrix} E & F \\ 0 & G \end{bmatrix}, \quad E \in \mathbb{C}^{k \times k}, G \in \mathbb{C}^{(n-k)\times(n-k)}.$$

Otherwise, we say that A is *unreduced*. We shall adopt the notation

$$A_s = \mathtt{QR}(A_0, \rho_0, \ldots, \rho_{s-1}), \quad s \geq 0,$$

to denote that A_s is obtained by means (2.1) applied to A_0 after s steps with shifts $\rho_0, \ldots, \rho_{s-1}$. The QR decomposition is not generally unique so that A_s is not univocally determined from the initial data and the selected shifts. However, essential uniqueness can be achieved by using an effectively eliminating QR factorization algorithm as defined in [18]. In this way the matrix $\mathtt{QR}(A_0, \rho_0, \ldots, \rho_{s-1})$ becomes (essentially) unique up to similarity by a unitary diagonal matrix.

Theorem 3.3 below describes the structure of the unitary matrix U_s, $s \geq 0$. This characterization mostly relies upon the banded form of the unitary factor computed by means of a QR factorization of A_s, $s \geq 0$, as stated in the next result.

LEMMA 3.1. *For any fixed $s \geq 0$ there exists a unitary CMV–like shaped matrix $Q \in \mathbb{C}^{n \times n}$ such that $Q^H A_s :\, = R$ is upper triangular, i.e., $A_s = QR$ gives a QR factorization of A_s. In particular, it satisfies*

$$Q(1:2j, 2(j+1):n) = 0, \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor - 1, \tag{3.4}$$

and

$$Q(2i+1:n, 1:2(i-1)+1) = 0, \quad 1 \leq i \leq \lfloor \frac{n+1}{2} \rfloor - 1. \tag{3.5}$$

Proof. Let us first recall that the matrix A_0 satisfies

$$\operatorname{rank}(A_0(2j+1:2(j+1), 2j:2j+1)) = 1, \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor - 1.$$

The property follows by direct inspection for the case $A_0 = \widehat{C}$ or by using Corollary 2.5. From the argument stated in Remark 2.8 we find that the rank constraint is propagated along the QR algorithm and, specifically, for any $s \geq 0$, we have

$$\operatorname{rank}(A_s(2j+1:2(j+1), 2j:2j+1)) \leq 1, \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor - 1, \tag{3.6}$$

where the equality holds if A_s is unreduced. A QR decomposition of the matrix A_s can be obtained in two steps. Assume that n is even for the sake of illustration. At the first step we determine Givens rotations

$$\mathcal{R}_1(\gamma_1,\sigma_1),\mathcal{R}_3(\gamma_3,\sigma_3)\ldots\mathcal{R}_{2\lfloor\frac{n}{2}\rfloor-1}(\gamma_{2\lfloor\frac{n}{2}\rfloor-1},\sigma_{2\lfloor\frac{n}{2}\rfloor-1}),$$

given as in (2.5) to annihilate, respectively, the entries of A_s in positions

$$(2,1),(4,2),\ldots,(2\lfloor\frac{n}{2}\rfloor,2\lfloor\frac{n}{2}\rfloor-2).$$

Let

$$\mathcal{L}=\mathcal{R}_1(\gamma_1,\sigma_1)^H\cdot\mathcal{R}_3(\gamma_3,\sigma_3)^H\cdots\mathcal{R}_{2\lfloor\frac{n+1}{2}\rfloor-1}(\gamma_{2\lfloor\frac{n}{2}\rfloor-1},\sigma_{2\lfloor\frac{n}{2}\rfloor-1})^H$$

be the unitary block diagonal matrix formed by these rotations. Due to the rank constraint (3.6) the zeroing process also introduces zero entries in positions

$$(4,3),\ldots,(2\lfloor\frac{n}{2}\rfloor,2\lfloor\frac{n}{2}\rfloor-1).$$

Then in the second step a sequence of Givens rotations

$$\mathcal{R}_2(\gamma_2,\sigma_2),\mathcal{R}_4(\gamma_4,\sigma_4)\ldots\mathcal{R}_{2\lfloor\frac{n}{2}\rfloor-2}(\gamma_{2\lfloor\frac{n}{2}\rfloor-2},\sigma_{2\lfloor\frac{n}{2}\rfloor-2})$$

is employed to make zero, respectively, the entries of $\mathcal{L}^HA_s$ in positions

$$(3,2),\ldots,(2\lfloor\frac{n}{2}\rfloor-1,2\lfloor\frac{n}{2}\rfloor-2).$$

This completes the reduction of A_s in upper triangular form. If we set

$$\mathcal{M}=\mathcal{R}_2(\gamma_2,\sigma_2)^H\cdot\mathcal{R}_4(\gamma_4,\sigma_4)^H\cdots\mathcal{R}_{2\lfloor\frac{n}{2}\rfloor-2}(\gamma_{2\lfloor\frac{n}{2}\rfloor-2},\sigma_{2\lfloor\frac{n}{2}\rfloor-2})^H,$$

then $\mathcal{M}$ is unitary block diagonal and

$$Q\colon=\mathcal{L}\cdot\mathcal{M}$$

is a unitary CMV–like shaped matrix. In particular, the pattern of its zero entries satisfies (3.4), (3.5) in accordance with Definition 2.3. The case of n odd is treated similarly. □

REMARK 3.2. *As noticed at the end of Subsection 2.1 the factorization of the unitary factor Q of a CMV–like shaped matrix as product of unitary block diagonal matrices is analogous with the decomposition* (2.4) *of unitary CMV matrices once we have replaced* (2.2) *with* (2.5).

Lemma 3.1 can be used to exploit the rank properties of the unitary matrices U_s, $s\geq 0$, obtained by updating the matrix $U_0=\widehat{U}$ under the QR process (2.1), (3.2), (3.3). We first analyze the case where $A_0=\widehat{C}$ is invertible.

THEOREM 3.3. *The matrices U_s, $s\geq 0$, generated as in* (3.3) *by the QR iteration* (2.1) *applied to an invertible $A_0=\widehat{C}$ satisfy*

$$\mathrm{rank}(U_s(1:2j,2(j+1):n))\leq 1,\quad 1\leq j\leq\lfloor\frac{n}{2}\rfloor-1,\ s\geq 0,$$

and, specifically,

$$U_s(1:2j,2(j+1):n)=B_s(1:2j,2(j+1):n),\quad 1\leq j\leq\lfloor\frac{n}{2}\rfloor-1,\ s\geq 0,$$

where

$$B_s := \frac{U_s \boldsymbol{w}_s \boldsymbol{z}_s^H U_s}{\boldsymbol{z}_s^H U_s \boldsymbol{w}_s - 1} = \left(\frac{\bar{p}_n}{\bar{p}_0}\right) \; U_s \boldsymbol{w}_s \boldsymbol{z}_s^H U_s = Q_s^H B_{s-1} Q_s, \quad s \geq 1, \tag{3.7}$$

is a rank one matrix.

Proof. The invertibility of A_0 implies the invertibility of A_s for any $s \geq 0$. Let $A_s = QR$ be a QR factorization of the matrix A_s, where Q is a unitary CMV–like shaped matrix determined as in Lemma 3.1. From

$$Q^H A_s = Q^H(U_s - \boldsymbol{z}_s \boldsymbol{w}_s^H) = Q^H U_s - Q^H \boldsymbol{z}_s \boldsymbol{w}_s^H = R \tag{3.8}$$

we obtain that

$$(Q^H A_s)^{-H} = Q^H (U_s - \boldsymbol{z}_s \boldsymbol{w}_s^H)^{-H} = R^{-H}.$$

Using the Sherman–Morrison formula [25] yields

$$Q^H(U_s + \frac{U_s \boldsymbol{w}_s \boldsymbol{z}_s^H U_s}{1 - \boldsymbol{z}_s^H U_s \boldsymbol{w}_s}) = R^{-H},$$

which gives

$$U_s = QR + \boldsymbol{z}_s \boldsymbol{w}_s^H = QR^{-H} - \frac{U_s \boldsymbol{w}_s \boldsymbol{z}_s^H U_s}{1 - \boldsymbol{z}_s^H U_s \boldsymbol{w}_s} = QR^{-H} + B_s$$

From $\det(A_s) = \det(A_0) = (-1)^n p_0/p_n$ and $\det(U_s) = \det(U_0) = (-1)^{n+1}$ it follows that

$$-\det(A_0^{-H}) = (-1)^{n+1} \bar{p}_n/\bar{p}_0 = \frac{(-1)^{n+1}}{\boldsymbol{z}_0^H U_0 \boldsymbol{w}_0 - 1} = \frac{(-1)^{n+1}}{\boldsymbol{z}_s^H U_s \boldsymbol{w}_s - 1}, \quad s \geq 0.$$

showing that $\bar{p}_n/\bar{p}_0 = -1/(1 - \boldsymbol{z}_s^H U_s \boldsymbol{w}_s)$.

Since R^{-1} is upper triangular we have that $R^{-1}Q^H = (QR^{-H})^H$ has the same lower staircase envelope as Q^H and, therefore, from Lemma 3.1 we conclude that

$$\operatorname{rank}(U_s(1:2j, 2(j+1):n)) \leq 1, \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor - 1, \; s \geq 0, \tag{3.9}$$

and, specifically,

$$U_s(1:2j, 2(j+1):n) = B_s(1:2j, 2(j+1):n), \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor - 1, \; s \geq 0,$$

□

The previous theorem opens the way for a condensed representation of each matrix A_s, $s \geq 0$, generated by (2.1) applied to an invertible initial matrix $A_0 = \widehat{C}$ in terms of a linear number of parameters including the entries of the vectors $\boldsymbol{f}_s = (\bar{p}_n/\bar{p}_0) U_s \boldsymbol{w}_s$ and $\boldsymbol{g}_s = U_s^H \boldsymbol{z}_s$. This representation will be exploited into more details in the next section. Here we conclude with two remarks on points of detail concerning the singular case and the computation of the vector $\boldsymbol{f}_s$.

Remark 3.4. *From* (3.8) *it follows that the relation*

$$Q^H U_s = (R^H + \boldsymbol{w}_s \boldsymbol{z}_s^H Q)^{-1}, \quad s \geq 0,$$

still holds in the degenerate case where A_0 and, a fortiori, A_s, $s \geq 0$, are singular. By using standard results about the inversion of rank–structured matrices [21] it is found that the inverse of a nonsingular lower triangular plus a rank–one matrix has a rank structure of order one in its strictly upper triangular part. Since Q is a unitary CMV–like shaped matrix, this property implies that (3.9) *is always satisfied independently of the invertibility of the starting matrix $A_0 = \widehat{C}$.*

REMARK 3.5. *The accuracy of computed eigenvalues generally depends on the magnitude of the generators employed to represent the matrix. In the almost singular case it is expected that the size of the entries of $\boldsymbol{f}_s$ can be very large and this can in principle deteriorate the quality of the approximations. However, it is worth noticing that all the entries of these vectors except the last two can be dynamically determined in a numerically robust way by considering the effects of zero–shifting at the early steps of the QR iteration* (2.1) *applied to a nonsingular $A_0 = \widehat{C}$. More specifically, let $A_s = \mathtt{QR}(A_0, 0, \ldots, 0)$, $1 \leq s \leq \lfloor \frac{n}{2} \rfloor - 1$, be the matrix generated by* (2.1) *applied to a nonsingular $A_0 = \widehat{C}$ after s iterations with zero shift. Then it is shown that*

$$\boldsymbol{f}_s(1:2s) = U_s(1:2s, 2s+1), \quad 1 \leq s \leq \lfloor \frac{n}{2} \rfloor - 1.$$

For the sake of brevity we omit the proof of this property but we provide a pictorial illustration by showing in Figure 3.1 the plot of the nonzero pattern of U_0, U_1, $U_{\lfloor \frac{n}{2} \rfloor - 2}$ and $U_{\lfloor \frac{n}{2} \rfloor - 1}$.

4. Fast Algorithms and Numerical Results. In this section we devise a fast adaptation of the QR iteration (2.1) applied to a starting invertible matrix $A_0 = \widehat{C} \in \mathbb{C}^{n\times n}$ given as in (2.3) by using the structural properties described above. First we describe the entrywise data–sparse representation of the matrices involved in the QR iteration and sketch the structured variant. Then we present the implementation of the resulting algorithm together with the results of extensive numerical experiments.

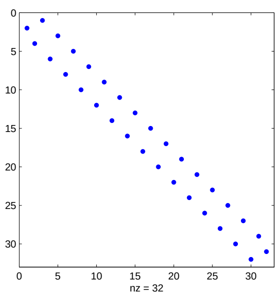
Our proposal is an explicit QR method applied to a permuted version of a companion matrix which, at each step, works on a condensed entrywise data–sparse representation of the matrix using $O(n)$ flops and $O(n)$ memory storage. Standard implementations of the QR eigenvalue algorithm for Hessenberg matrices are based on implicit schemes relying upon the implicit Q theorem [25]. A derivation of the implicit Q theorem for staircase matrices in block upper Hessenberg form requires some cautions [37] and will be addressed elsewhere.

4.1. Data–Sparse representation and Structured QR. Let us start by observing that each matrix A_s, $s \geq 0$, generated by (2.1) can be represented by means of a banded matrix $\widehat{A}_s$ which contains the entries of A_s within the staircase pattern and of four vectors to represent the rank two structure in the upper triangular portion of A_s above the staircase profile. Using the following sparse data representation we need to store just $O(n)$ entries:

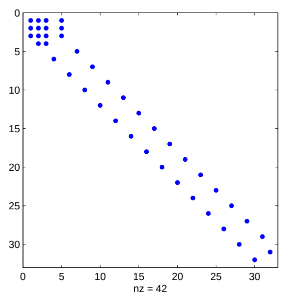
1. the nonzero entries of the banded matrix $\widehat{A}_s = (\hat{a}^{(s)}_{i,j}) \in \mathbb{C}^{n\times n}$ obtained from A_s according to

$$\hat{a}^{(s)}_{i,j} = \begin{cases} a^{(s)}_{i,j}, \text{ if } \max\{1, 2\lfloor\frac{i+1}{2}\rfloor - 2\} \leq j \leq \min\{2\lfloor\frac{i+1}{2}\rfloor + 2, n\}, i = 1\ldots, n; \\ 0, \text{ elsewhere;} \end{cases}$$

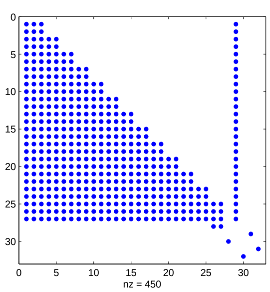
2. the vectors $\boldsymbol{z}_s = (z^{(s)}_i), \boldsymbol{w}_s = (w^{(s)}_i) \in \mathbb{C}^n$ and $\boldsymbol{f}_s :\ = (\bar{p}_n/\bar{p}_0)U_s\boldsymbol{w}_s, \boldsymbol{f}_s = (f^{(s)}_i)$, and $\boldsymbol{g}_s :\ = U_s^H\boldsymbol{z}_s, \boldsymbol{g}_s = (g^{(s)}_i)$.



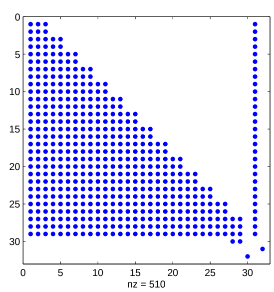
(a) U_0



(b) U_1



(c) $U_{\lfloor \frac{n}{2} \rfloor - 2}$



(d) $U_{\lfloor \frac{n}{2} \rfloor - 1}$

Figure 3.1: Sparsity pattern of U_j with $j = 1, 2, \lfloor \frac{n}{2} \rfloor - 2, \lfloor \frac{n}{2} \rfloor - 1$.

The nonzero pattern of the matrix $\widehat{A}_s$ looks as below:

$$\widehat{A}_s = \begin{bmatrix} \star & \star & \star & \star & & & & \\ \star & \star & \star & \star & & & & \\ & \star & \star & \star & \star & \star & & \\ & \star & \star & \star & \star & \star & & \\ & & & \star & \star & \star & \star & \star \\ & & & \star & \star & \star & \star & \star \\ & & & & & \star & \star & \star \\ & & & & & \star & \star & \star \end{bmatrix}, \qquad (n = 2k),$$

or

$$\widehat{A}_s = \begin{bmatrix} \star & \star & \star & \star & & & & \\ \star & \star & \star & \star & & & & \\ & \star & \star & \star & \star & \star & & \\ & \star & \star & \star & \star & \star & & \\ & & & \star & \star & \star & \star & \\ & & & \star & \star & \star & \star & \\ & & & & & \star & \star & \end{bmatrix}, \qquad (n = 2k-1).$$

From (3.2) and (3.7) we find that the entries of the matrix $A_s = (a_{i,j}^{(s)})$ can be expressed in terms of elements of this data set as follows:

$$a_{i,j}^{(s)} = \begin{cases} f_i^{(s)} \bar{g}_j{}^{(s)} - z_i^{(s)} \bar{w}_j{}^{(s)}, \text{ if } j \geq 2\lfloor \frac{i+1}{2} \rfloor + 3, \ 1 \leq i \leq 2\lfloor \frac{n+1}{2} \rfloor - 4; \\ \widehat{a}_{i,j}^{(s)}, \text{ elsewhere.} \end{cases} \tag{4.1}$$

The next procedure performs a structured variant of the QR iteration (2.1) applied to an initial matrix $A_0 = \widehat{C} \in \mathbb{C}^{n \times n}$ given as in (2.3).

Procedure Fast_QR
Input: $\widehat{A}_s$, $\boldsymbol{z}_s$, $\boldsymbol{w}_s$, $\boldsymbol{f}_s$, $\boldsymbol{g}_s$;
Output: $\widehat{A}_{s+1}$, ρ_s, $\boldsymbol{z}_{s+1}$, $\boldsymbol{w}_{s+1}$, $\boldsymbol{f}_{s+1}$, $\boldsymbol{g}_{s+1}$;
1. Compute the shift ρ_s.
2. Find the factored form of the matrix Q_s such that
   $Q_s^H(A_s - \rho_s I) = R_s, \quad R_s$ upper triangular,
   where A_s is represented via (4.1).
3. Determine $\widehat{A}_{s+1}$ from the entries of $A_{s+1} = Q_s^H A_s Q_s$.
4. Evaluate $\boldsymbol{z}_{s+1} = Q_s^H \boldsymbol{z}_s$, $\boldsymbol{w}_{s+1} = Q_s^H \boldsymbol{w}_s$, $\boldsymbol{f}_{s+1} = Q_s^H \boldsymbol{f}_s$, $\boldsymbol{g}_{s+1} = Q_s^H \boldsymbol{g}_s$.

The factored form of Q_s makes it possible to execute the steps $2, 3$ and 4 simultaneously by improving the efficiency of computation. The matrix A_s is represented by means of four vectors and a diagonally structured matrix $\widehat{A}_s$ encompassing the band profile of A_s. This matrix could be stored in a rectangular array but for the sake of simplicity in our implementation we adopt the MatLab sparse matrix format. Due to the occurrences of deflations the QR process is applied to a principal submatrix of A_s starting at position $pst + 1$ and ending at position $n - qst$, where $pst = qst = 0$ at beginning.

At the core of **Fast_QR** (steps 2–4) there is the following scheme, where for notational simplicity we set $pst = qst = 0$, n even, denote $T_s = \boldsymbol{f}_s \boldsymbol{g}_s^H - \boldsymbol{z}_s \boldsymbol{w}_s^H$, and we omit the subscript s from $\widehat{A}_s$. In the case of negative or above n indices we mean 1 or n respectively.

for $j = 1{:}\, n/2 - 1$

1. compute the QR factorization of $\widehat{A}(2j{:}2j+2, 2j{:}2j+2) - \rho_s I_3 = QR$;

2. update the matrix $\widehat{A}$ and the vectors $\boldsymbol{z}_s$, $\boldsymbol{w}_s$, $\boldsymbol{f}_s$, $\boldsymbol{g}_s$ by computing:

$$\widehat{A}(2j{:}2(j+1), 2(j-1){:}2(j+1)) = Q^H \left[\widehat{A}(2j{:}2(j+1), 2(j-1){:}2(j+1))\right],$$

$$\widehat{A}(2j{:}2(j+1), 2j+3{:}2(j+2)) = Q^H \left[\frac{T_s(2j, 2j+1{:}2j+2)}{\widehat{A}(2j+1{:}2(j+1), 2j+3{:}2(j+2))}\right],$$

$$\widehat{A}(2j-2{:}2(j+2), 2j{:}2(j+1)) = \widehat{A}(2j-2{:}2(j+2), 2j{:}2(j+1))\, Q,$$

$$\widehat{A}(2j-3, 2j{:}2(j+1)) = \left[\widehat{A}(2j-3, 2j) | T_s(2j-3, 2j+1, 2(j+1))\right] Q,$$

$$\boldsymbol{f}_{s+1}(2j{:}2j+2) = Q^H \boldsymbol{f}_s(2j{:}2j+2),\ \boldsymbol{z}_{s+1}(2j{:}2j+2) = Q^H \boldsymbol{z}_s(2j{:}2j+2),$$

$$\boldsymbol{g}_{s+1}(2j{:}2j+2) = Q^H \boldsymbol{g}_s(2j{:}2j+2),\ \boldsymbol{w}_{s+1}(2j{:}2j+2) = Q^H \boldsymbol{w}_s(2j{:}2j+2).$$

end

From a computational viewpoint this scheme basically amounts to perform two moltiplications of a matrix of size 8×3 by a 3×3 matrix at the cost of $2 \cdot 72$ flops, where flop stands for an axpy operation like $e = c + a * b$. Also observe that the matrix $\widehat{A}_s$ is slightly enlarged during the iterative process by reaching a maximal size of $n \times 8$.

4.2. Implementation Issues and Numerical Results. Shifting and deflation techniques are important concepts in the practical implementation of the QR method. The computation of the shift ρ_s at the first step of **Fast_QR** can be carried out by several strategies [25]. In our implementation we employ the Wilkinson idea by choosing as a shift one of the eigenvalues of the trailing 2-by-2 submatrix of $A_s(pst{+}1 : n - qst, pst + 1 : n - qs)$ (the one closest to the tailing entry).

For staircase matrices deflation can occur along the vertical or horizontal edges of the co–diagonal blocks. In our implementation we apply the classical criterion for deflation by comparing the entries located on these edges with the neighborhood diagonal entries of the matrix, that is we have a deflation if either

$$\|\widehat{A}_s(2j+1 : 2(j+1), 2j)\| \le \varepsilon\,(|\widehat{A}_s(2j, 2j)| + |\widehat{A}_s(2j+1, 2j+1)|,$$

or

$$\|\widehat{A}_s(2(j+1), 2j : 2j+1)\| \le \varepsilon\,(|\widehat{A}_s(2j+1, 2j+1)| + |\widehat{A}_s(2(j+1), 2(j+1))|.$$

If the test is fulfilled then the problem splits in two subproblems that are treated individually. Incorporating the Wilkinson shifting and the deflation checks within the explicit shifted QR method **Fast_QR** and implementing a step of the QR iteration according to the scheme above, yields our proposed fast CMV–based eigensolver for companion matrices. The algorithm has been implemented in MatLab and tested on several examples. This implementation can be obtained from the authors upon request.

For an input companion matrix expressed as a rank–one correction of a unitary CMV–like shaped matrix the Wilkinson strategy ensures zero shifting at the early iterations. This has the advantage of allowing a check on the construction of the vector $\boldsymbol{f}_s$ as described in Remark 3.5. Additionally, it has been observed experimentally that this shift strategy is important for the correct fill–in within the band profile of A_s as it causes a progressive fill–in of the generators and of the band profile by ensuring that the fundamental condition (3.6) is numerically satisfied.

In order to check the accuracy of the output we compare the computed approximations with the ones returned by the internal function `eig` applied to the initial companion matrix $C = C(p) \in \mathbb{C}^{n\times n}$ with the balance option on. Specifically, we match the two lists of approximations and then find the average absolute error $\mathrm{err}_{\mathbf{FastQR}} = \sum_{j=1}^{n} err_j/n$. For a backward stable algorithm in the light of the classical perturbation results for eigenvalue computation [25] we know that this error would be of the order of $\|\Delta C\|_\infty \, \mathcal{K}_\infty(V)\, \varepsilon$, where $\|\Delta C\|_\infty$ is the backward error, $\mathcal{K}_\infty(V) = \|V\|_\infty \cdot \|V^{-1}\|_\infty$ is the condition number of V, the eigenvector matrix of C and ε denotes the machine precision.

A backward stability analysis of the customary QR eigenvalue algorithm is performed in [34] by showing that $\|\Delta C\|_F \leq cn^3\|C\|_F$ for a small integer constant c. A partial extension of this result to certain fast adaptations of the QR algorithm for rank–structured matrices is provided in [20] by replacing $\|C\|_F$ with a measure of the magnitude of the generators. The numerical experience reported in [10] further supports this extension. In the present case, considering the infinity norm, we find that

$$\begin{aligned}\|C\|_\infty = \|A_0\|_\infty &\leq \|\widehat{A}_0\|_\infty + \|\boldsymbol{f}_0\|_\infty\|\boldsymbol{g}_0\|_\infty + \|\boldsymbol{w}_0\|_\infty\|\boldsymbol{z}_0\|_\infty \\ &= \|\widehat{A}_0\|_\infty + \|\boldsymbol{f}_0\|_\infty + \|\boldsymbol{w}_0\|_\infty.\end{aligned}$$

The parameter $\sigma = \bar{p}_n/\bar{p}_0$ in the starting representation via generators is incorporated into the vector $\boldsymbol{f}_0$, leading to a vector whose entries depend on the ratios $\pm p_j/p_0$. Viceversa, the entries of vector $\boldsymbol{w}_0$, depend on the ratios $\pm p_j/p_n$. Backward stability with respect to the input data $\widehat{A}_0$, $\boldsymbol{f}_0$, $\boldsymbol{g}_0$, $\boldsymbol{w}_0$ and $\boldsymbol{z}_0$ would imply that the maximum expected absolute error depends on

$$\mathtt{mee} = \left(\|\widehat{A}_0\|_\infty + \|\sigma U_0 \boldsymbol{w}_0\|_\infty + \|\boldsymbol{w}_0\|_\infty\right) \mathcal{K}_\infty(V)\, \varepsilon.$$

This quantity can be really larger than $\|C\|_\infty \mathcal{K}_\infty(V)\, \varepsilon$ especially when the coefficients of the polynomial $p(z)$ are highly unbalanced. A favorable case is when the starting polynomial is (anti)palindromic ($\sigma = \pm 1$) and it can be used for accuracy comparisons with standard $O(n^3)$ matrix methods. In order to reduce the effects of the magnitude of the generators on the accuracy of computed results we implement a scaling strategy as the one described in [16]. Specifically, for any input polynomial $p(z) = \sum_{i=0}^{n} p_i z^i$ we define

$$p_s(z) = p(z\cdot\alpha^s) = p_0^{(s)} + p_1^{(s)} z + \ldots + p_n^{(s)} z^n, \quad p_j^{(s)} = \alpha^{js} p_j,\ 0 \leq j \leq n,$$

and then we determine the value of the scaling parameter $s \in \mathbb{Z}$ so that

$$\chi(\boldsymbol{p}, s) = \frac{\max |p_j^{(s)}|}{\min |p_j^{(s)}|}$$

is as small as possible. In practice we restrict s over a small segment around the origin, say $-6 \le s \le 6$, and we take $\alpha = 2$ in accordance with the extensive experimentation carried on in [36]. The scaling strategy is applied whenever the presence of a high σ will alter the original conditioning of the problem, that is when $|\bar{p}_n| > |\bar{p}_0|$.

Our implementation reports as output the value of `wer` = `err`/`mee`, **with the aim of estimating relation between** $\|\Delta C\|_\infty$ **and** $\|C\|_\infty$. In accordance with our claim this quantity should be bounded by a small multiple of n^3, in practical situation this quantity is never larger than 1. As a measure of efficiency of the algorithm we also determine $\mathtt{it_{av}}$, the average number of QR steps per eigenvalue which shows that the cost of this fast adaptation of QR iterates has indeed a quadratic cost for approximating all the eigenvalues, since $\mathtt{it_{av}}$ is always between 2 and 5.

We have performed many numerical experiments with real polynomials of both small and large degree. Moreover, to support our expectation about the very good behavior of the method when the polynomial has balanced coefficients we consider several cases where the input polynomial is (anti)palindromic in such a way that $\|\boldsymbol{f}_0\| = \|\sigma U_0 \boldsymbol{w}_0\|_\infty = \|\boldsymbol{w}_0\|_\infty$. Our test suite consists of the following polynomials already used as tests suite by other authors:

- (P1) $p(z) = 1 + (\frac{n}{n+1} + \frac{n+1}{n}) z^n + z^{2n}$ [9]. The zeros can be explicitly determined and lie on two circles centered at the origin that are poorly separated.
- (P2) $p(z) = \frac{1}{n}\left(\sum_{j=0}^{n-1}(n+j)z^j + (n+1)z^n + \sum_{j=0}^{n-1}(n+j)z^{2n-j}\right)$ [12]. This is another test problem for spectral factorization algorithms.
- (P3) $p(z) = (1-\lambda)z^{n+1} - (\lambda+1)z^n + (\lambda+1)z - (1-\lambda)$ [1]. This family of antipalindromic polynomials arises in the context of a boundary–value problem whose eigenvalues coincide with the zeros of an entire function related with $p(z)$.
- (P4) A collection of small–degree polynomials [35]:
  1. the Bernoulli polynomial $p(z) = \sum_{j=0}^{n} \left(\begin{array}{c} n \\ j \end{array}\right) b_{n-j} z^j$, where b_j are the Bernoulli numbers;
  2. the Chebyshev polynomial of first kind;
  3. the partial sum of the exponential $p(z) = \sum_{j=0}^{n} (2z)^j / j!$.
- (P5) Polynomial $p(z) = \sum_{j=0}^{n} p_j z^j$ where the coefficients p_j are drawn from the uniform distribution in $[0, 1]$.
- (P6) Polynomials $p(z) = \sum_{j=0}^{n} p_j z^j$ with coefficients of the form $p_j = a_j \times 10^{e_j}$, where a_j and e_j are drawn from the uniform distribution in $[-1, 1]$ and $[-3, 3]$, respectively. These polynomials were proposed in [27] for testing purposes.
- (P7) The symmetrized version of the previous polynomials, that is, $p(z) = s(z)s(z^{-1})z^n$ where $s(z) = \sum_{j=0}^{n} s_j z^j$ with coefficients of the form $s_j = a_j \times 10^{e_j}$ and $a_j \in [-1, 1]$ and $e_j \in [-3, 3]$.

For the sake of illustration in Figure 4.1 and 4.2 we also display the distribution of the zeros computed by our routine and by the MatLab function `eig` applied to polynomials in the class $P2$ and $P3$, respectively. We see that the ticks are undistinguishable since our algorithms is very accurate.

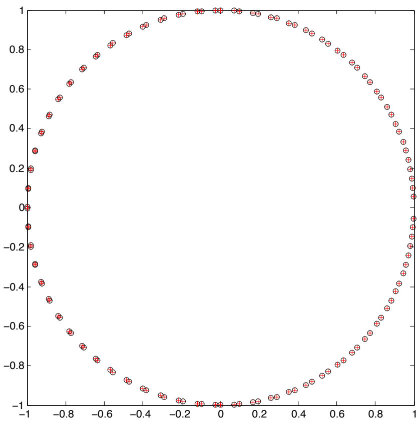
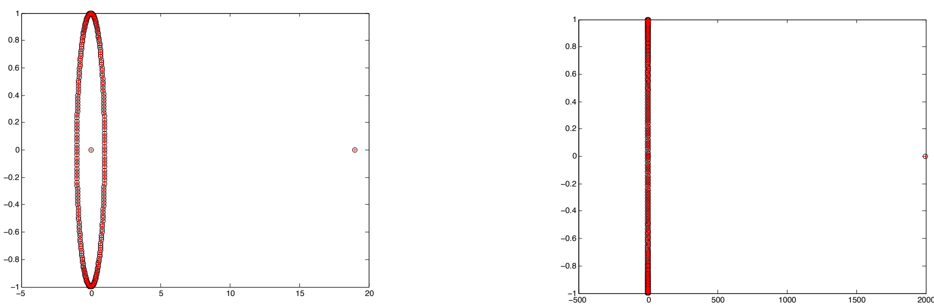


Figure 4.1: Distribution of the zeros computed by our routine (plus) and `eig` (circles) for the polynomial in the class $P2$ of degree $n = 128$.



(a) (b)

Figure 4.2: Distribution of the zeros computed by our routine (plus) and `eig` (circles) for the polynomials in the class $P3$ of degree $n = 128$ with $\lambda = 0.9$ (a) and $\lambda = 0.999$ (b).

Table 4.1 shows the numerical results for the first three sets of palyndromic polynomials. Together with the two values $\texttt{mee}/\varepsilon$ and `wer` we report the average absolute error of the **FastQR** algorithm and for comparison also the average absolute error of the algorithm, denoted in our tests as $\mathbf{B^2EG^2}$, as initially proposed in [5] and then improved in [11] by exploiting the technique of compression of the generators. We choose to compare our solver with the fast algorithm proposed in [5, 11] because in [3] it has been pointed out as "one of the best of the structured codes that have been proposed so far". To compare the two methods also in terms of execution time, for polynomials of large degree we also report the parameter $\texttt{T}_{\texttt{ratio}}$ given by the ratio between the time required by the $\mathbf{B^2EG^2}$ method and our method. We see that our algorithm on these examples is always very accurate, also on cases such as polynomial of large degree of type $P1$ where a certain degeneration of the accuracy can be observed in the performance of method $\mathbf{B^2EG^2}$. Observing the values of $\texttt{T}_{\texttt{ratio}}$ it turns

Test Set	n	`mee`$/\varepsilon$	$\mathtt{err}_{\mathbf{FastQR}}$	$\mathtt{err}_{\mathbf{B^2EG^2}}$	`wer`	$\mathtt{it}_{\mathtt{av}}$	$\mathtt{T}_{\mathtt{ratio}}$
	64	4.14e+04	4.13e-14	7.29e-13	4.50e-03	4.53	4.20
	128	1.65e+05	9.23e-14	5.57e-12	2.52e-03	4.52	4.85
$P1$	256	6.57e+05	3.00e-13	2.92e-11	2.06e-03	4.52	4.78
	512	2.62e+06	1.01e-12	1.65e-10	1.73e-03	4.51	4.63
	1024	1.05e+07	2.47e-12	1.18e-09	1.06e-03	4.51	4.63
	64	9.59e+03	5.80e-15	9.75e-15	2.72e-03	3.34	4.51
	128	2.73e+04	8.55e-15	2.59e-14	1.41e-03	3.22	4.36
$P2$	256	7.76e+04	1.38e-14	7.08e-14	7.98e-04	3.10	4.82
	512	2.20e+05	3.17e-14	3.75e-13	6.48e-04	3.03	3.81
	1024	6.23e+05	3.72e-14	1.43e-12	2.69e-04	2.96	3.31
	64	1.10e+04	4.70e-15	6.68e-15	1.93e-03	2.94	6.17
	128	2.20e+04	6.72e-15	1.23e-14	1.38e-03	2.67	7.67
$P3$	256	4.41e+04	1.24e-14	2.97e-14	1.27e-03	2.57	5.87
$(\lambda = 0.9)$	512	8.83e+04	2.59e-14	7.02e-14	1.32e-03	2.53	5.99
	1024	1.77e+05	3.85e-14	1.87e-13	9.82e-04	2.51	4.93
	64	1.08e+06	5.46e-15	1.01e-14	2.28e-05	3.03	6.66
	128	2.16e+06	6.03e-15	1.68e-14	1.26e-05	2.71	6.65
$P3$	256	4.34e+06	8.25e-15	3.52e-14	8.57e-06	2.58	6.04
$(\lambda = 0.999)$	512	8.68e+06	9.95e-15	7.37e-14	5.16e-06	2.54	5.78
	1024	1.74e+07	1.57e-14	1.88e-13	4.06e-06	2.52	4.59

Table 4.1: Numerical results for the sets $P1$, $P2$ and $P3$ of (anti)palindromic polynomials. The degree of the polynomials considered for these tests is $2n$.

out that our method is at least 3.31 times faster, and in most cases, 4 times faster than the one in [11].

Table 4.2 shows the numerical results for the small degree polynomials $P4$. For the sake of illustration in Figure 4.3 and 4.4 we also display the distribution of the zeros computed by our routine and the MatLab function `eig` applied to polynomials in the class $P4(1-2)$ and $P4(3)$, respectively.

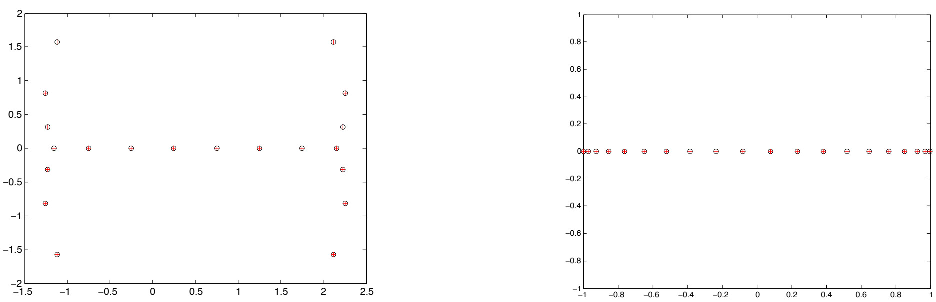
In the Chebyshev case the use of the scaling technique has the effect of reducing the magnitude of the generators and allows to improve the accuracy of computed results that would be worse if this technique was not applied.

Table 4.3 gives the numerical results for the polynomials with random coefficients of type $P5$, $P6$ and $P7$. Here we report for `mee`$/\varepsilon$ the min/max range and for the other columns the maximum value of the data output variables over fifty experiments. We note that among the fifty random tests we have either mildly or seriously ill conditioned instances and this is a very hard test for our method since scaling is not usually effective on these instances. In general the algorithm $\mathbf{B^2EG^2}$ has a better accuracy requiring however much more time. The accuracy obtained by our method is however still largely within the conjectured bounds, in particular `wer` is far away from the theoretical bound of n^3.

Finally in figure 4.5 we compare our method (asterisks) and the $\mathbf{B^2EG^2}$ method (circles) on the basis of execution time for computing the roots of the polynomial $2^n\, z^n + 1$ for values of n ranging from 16 to 512. This plot confirms the fact that both

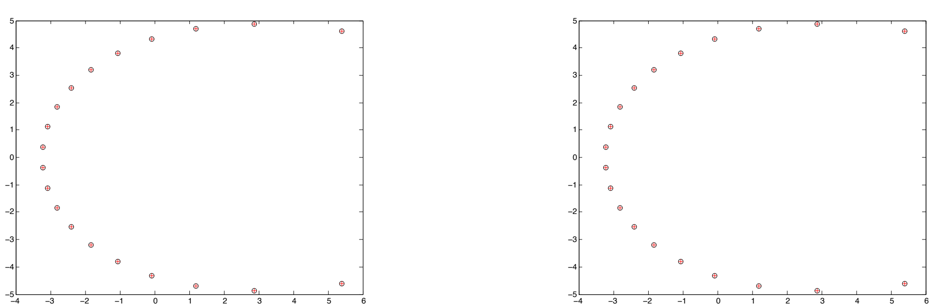
Test Set	n	$\mathtt{mee}/\varepsilon$	$\mathtt{err}_{\mathbf{FastQR}}$	$\mathtt{err}_{\mathbf{B^2EG^2}}$	$\mathtt{wer}$	$\mathtt{it}_{\mathtt{av}}$
	10	5.01e+05	3.38e-14	7.75e-15	3.04e-04	3.50
$P4(1)$	20	1.34e+13	4.60e-13	6.54e-11	1.54e-10	3.50
	30	5.94e+25	3.92e-11	2.76e-04	2.97e-21	3.77
	10	1.83e+05	6.22e-15	6.29e-15	1.54e-04	3.20
$P4(2)$	20	1.06e+12	1.79e-11	1.36e-10	7.64e-08	3.30
	30	8.61e+18	8.10e-08	7.10e-09	4.24e-11	3.37
	10	2.93e+08	5.26e-15	3.37e-12	8.07e-08	3.20
$P4(3)$	20	9.83e+25	6.94e-12	2.74e-01	3.18e-22	3.35
	30	8.49e+47	3.75e-08	6.83e+00	1.99e-40	3.30

Table 4.2: Numerical results for the sets $P4(1-3)$. For these tests we do not report the time comparison, in fact for small examples it is not significant (however the execution times were lower for the **FastQR** method).



(a) (b)

Figure 4.3: Distribution of the zeros of Bernoulli and Chebyshev polynomial of degree 20 computed by our routine (plus) and `eig` (circles).



(a) (b)

Figure 4.4: Distribution of the zeros of truncated Taylor series of e^{2z} of degree 20 and 30 computed by our routine (plus) and `eig` (circles).

Test Set	n	mee/ε	$\mathtt{err}_{\mathbf{FastQR}}$	$\mathtt{err}_{\mathbf{B^2EG^2}}$	wer	$\mathtt{it}_{\mathtt{av}}$	$\mathtt{T}_{\mathtt{ratio}}$
$P5$	64	2.09e+03- 1.89e+05	5.73e-15	5.51e-15	1.75e-03	3.47	4.73
	128	6.02e+03- 3.24e+06	2.94e-14	1.03e-14	6.54e-04	3.34	4.51
	256	2.04e+04- 4.47e+06	7.75e-14	1.97e-14	3.66e-04	3.18	4.36
	512	1.33e+05- 3.33e+08	6.20e-13	4.46e-14	1.00e-04	3.05	4.11
$P6$	16	7.47e+01- 5.40e+11	3.58e-09	5.19e-11	2.60e-05	3.47	5.00
	32	5.22e+02- 9.79e+12	5.72e-10	2.38e-09	2.64e-03	3.64	4.80
	64	7.46e+03- 3.37e+13	3.82e-08	2.89e-11	5.03e-09	3.55	4.78
	128	1.28e+05- 3.11e+13	6.18e-09	7.18e-11	1.01e-03	3.38	4.51
$P7$	16	3.90e+02- 1.89e+20	1.37e-03	6.95e-04	1.40e-04	3.55	5.13
	32	7.44e+04- 3.64e+17	3.29e-04	1.00e-04	1.20e-02	3.59	4.69
	64	6.87e+07- 5.02e+21	1.91e+00	1.08e-03	5.75e-04	3.41	4.65
	128	9.69e+06- 4.02e+27	4.99e+00	2.19e-03	2.95e-01	3.29	4.32

Table 4.3: Numerical results for the sets $P5$, $P6$ and $P7$. For simplicity we denoted here by $\mathtt{err}_{\mathtt{FastQR}}$, $\mathtt{err}_{B^2EG^2}$ the average error obtained over all the fifty random tests. Also $\mathtt{it}_{\mathtt{av}}$, wer and $\mathtt{T}_{\mathtt{ratio}}$ are the values obtained averaging the correspondent values over the corresponding fifty random values. Note that the polynomials of type P7 are palindromic and hence their degree is $2n$.

methods exhibit a quadratic time complexity and that the **FastQR** is faster than the method proposed in [5, 11].

5. Conclusion and Future Work. In this paper we have presented a novel fast QR–based eigensolver for companion matrices exploiting the structured technology for CMV–like representations. To our knowledge this is the first numerically reliable fast adaptation of the QR algorithm for perturbed unitary matrices which makes use of only four vectors to express the rank structure of the matrices generated under the iterative process. As a result, we obtain a data sparse parametrization of these matrices which at the same time is able to capture the structural properties of the matrices and yet to be sufficiently easy to manipulate and update for computations. The numerical experience is promising and confirms that the proposed approach performs faster while achieving a comparable accuracy at least under some restrictions on the magnitude of the starting generators. An approach that is amenable to circunvent these restrictions is presented in Remark 3.5 where an alternative construction of the vector $\boldsymbol{f}_0$ is shown. While it is clearly speculative at this point in development, this idea surely presents some interesting possibilities for future work. Another important issue concerns with the extension of this method to deal with matrix polynomials and generalized linearizations using block companion forms or diagonal plus small rank matrices.

Acknoledgments. We would like to thank Paola Boito for providing us with the MatLab code of the algorithm presented in [11]. The authors also thank the referees whose constructive comments have improved the quality of this work.

REFERENCES

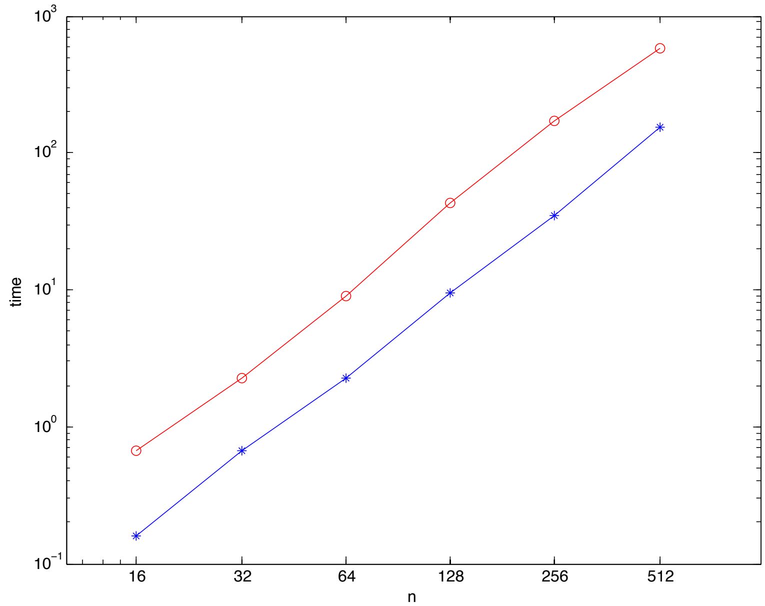


Figure 4.5: Plot in logarithmic scale of the time required by the **FastQR** algorithm (asterisks) and by the $\mathbf{B^2EG^2}$ algorithm (circles) for the polynomial $2^n z^n + 1$, for different values of the degree n.

[1] T. Aktosun, D. Gintides, and V. G. Papanicolaou. The uniqueness in the inverse problem for transmission eigenvalues for the spherically symmetric variable-speed wave equation. *Inverse Problems*, 27(11):115004, 17, 2011.

[2] P. Arbenz and G. H. Golub. Matrix shapes invariant under the symmetric QR algorithm. *Numer. Linear Algebra Appl.*, 2(2):87–93, 1995.

[3] J. L. Aurentz, R. Vandebril, and D. S. Watkins. Fast computation of the zeros of a polynomial via factorization of the companion matrix. *SIAM J. Sci. Comput.*, 35(1):A255–A269, 2013.

[4] R. Bevilacqua, G. M. Del Corso, and L. Gemignani. Compression of unitary rank–structured matrices to CMV-like shape with an application to polynomial rootfinding. *ArXiv e-prints*, July 2013.

[5] D. A. Bini, P. Boito, Y. Eidelman, L. Gemignani, and I. Gohberg. A fast implicit QR eigenvalue algorithm for companion matrices. *Linear Algebra Appl.*, 432(8):2006–2031, 2010.

[6] D. A. Bini, F. Daddi, and L. Gemignani. On the shifted QR iteration applied to companion matrices. *Electron. Trans. Numer. Anal.*, 18:137–152 (electronic), 2004.

[7] D. A. Bini, Y. Eidelman, L. Gemignani, and I. Gohberg. Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices. *SIAM J. Matrix Anal. Appl.*, 29(2):566–585, 2007.

[8] D. A. Bini, Y. Eidelman, L. Gemignani, and I. Gohberg. The unitary completion and QR iterations for a class of structured matrices. *Math. Comp.*, 77(261):353–378, 2008.

[9] D. A. Bini, G. Fiorentino, L. Gemignani, and B. Meini. Effective fast algorithms for polynomial spectral factorization. *Numer. Algorithms*, 34(2-4):217–227, 2003. International Conference on Numerical Algorithms, Vol. II (Marrakesh, 2001).

[10] P. Boito, Y. Eidelman, and L. Gemignani. Implicit QR for companion-like pencils. Technical report, arXiv:1401.5606, 2014.

[11] P. Boito, Y. Eidelman, L. Gemignani, and I. Gohberg. Implicit QR with compression. *Indagationes Mathematicae*, 23(4):733 – 761, 2012. Special Issue: Indagationes Mathematicae honoring Israel Gohberg.

[12] A. Böttcher and M. Halwass. Wiener-Hopf and spectral factorization of real polynomials by Newton's method. *Linear Algebra Appl.*, 438(12):4760–4805, 2013.

[13] A. Bunse-Gerstner and L. Elsner. Schur parameter pencils for the solution of the unitary eigenproblem. *Linear Algebra Appl.*, 154/156:741–778, 1991.

[14] M. J. Cantero, L. Moral, and L. Velázquez. Five-diagonal matrices and zeros of orthogonal polynomials on the unit circle. *Linear Algebra Appl.*, 362:29–56, 2003.

[15] M. J. Cantero, L. Moral, and L. Velázquez. Minimal representations of unitary operators and orthogonal polynomials on the unit circle. *Linear Algebra Appl.*, 408:40–65, 2005.

[16] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu. A fast QR algorithm for companion matrices. In *Recent advances in matrix and operator theory*, volume 179 of *Oper. Theory Adv. Appl.*, pages 111–143. Birkhäuser, Basel, 2008.

[17] F. De Terán, F. M. Dopico, and J. Pérez. Backward stability of polynomial root-finding using Fiedler companion matrices. submitted, 2014.

[18] S. Delvaux and M. Van Barel. Rank structures preserved by the QR-algorithm: the singular case. *J. Comput. Appl. Math.*, 189(1-2):157–178, 2006.

[19] L. Dieci and T. Eirola. On smooth decompositions of matrices. *SIAM J. Matrix Anal. Appl.*, 20(3):800–819, 1999.

[20] Y. Eidelman, L. Gemignani, and I. Gohberg. Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations. *Numer. Algorithms*, 47(3):253–273, 2008.

[21] Y. Eidelman, I. Gohberg, and I. Haimovici. *Separable type representations of matrices and fast algorithms. Vol. 2*, volume 235 of *Operator Theory: Advances and Applications*. Birkhäuser/Springer Basel AG, Basel, 2014. Eigenvalue method.

[22] M. Fiedler and T. L. Markham. Completing a matrix when certain entries of its inverse are specified. *Linear Algebra Appl.*, 74:225–237, 1986.

[23] R. Fourer. Staircase matrices and systems. *SIAM Rev.*, 26(1):1–70, 1984.

[24] K. Frederix, S. Delvaux, and M. Van Barel. An algorithm for computing the eigenvalues of block companion matrices. *Numer. Algorithms*, 62(2):261–287, 2013.

[25] 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.

[26] W. B. Gragg. The QR algorithm for unitary Hessenberg matrices. *J.Comput. Appl. Math.*, 16:1–8, 1986.

[27] M. A. Jenkins and J. F. Traub. Principles for testing polynomial zerofinding programs. *ACM Trans. Math. Software*, 1(1):26–34, 1975.

[28] R. Killip and I. Nenciu. CMV: the unitary analogue of Jacobi matrices. *Comm. Pure Appl. Math.*, 60(8):1148–1188, 2007.

[29] J. M. McNamee. *Numerical methods for roots of polynomials. Part I*, volume 14 of *Studies in Computational Mathematics*. Elsevier B. V., Amsterdam, 2007.

[30] C. Moler. Roots - of polynomials, that is. Cleve's Corner, 1991.

[31] C. Moler. Fiedler companion matrix. Cleve's Corner, 2013.

[32] V. Y. Pan. Solving a polynomial equation: some history and recent progress. *SIAM Rev.*, 39(2):187–220, 1997.

[33] B. N. Parlett. The Fiedler companion matrix. Book of Abstracts, Householder Symposium XIX June 8-13, Spa Belgium, 2014.

[34] F. Tisseur. Backward stability of the QR algorithm. Technical Report 239, UMR 5585 Lyon Saint-Etienne, 1996.

[35] K. C. Toh and L. N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. *Numer. Math.*, 68(3):403–425, 1994.

[36] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix. Implicit double shift QR-algorithm for companion matrices. *Numerische Mathematik*, 116(2):177–212, 2010.

[37] R. Vandebril. Chasing bulges or rotations? A metamorphosis of the QR-algorithm. *SIAM J. Matrix Anal. Appl.*, 32(1):217–247, 2011.

[38] R. Vandebril and G.M. Del Corso. An implicit multishift QR-algorithm for hermitian plus low rank matrices. *SIAM Journal on Scientific Computing*, 32(4):2190–2212, 2010.

[39] R. Vandebril, M. Van Barel, and N. Mastronardi. *Matrix computations and semiseparable matrices. Vol. II.* Johns Hopkins University Press, Baltimore, MD, 2008. Eigenvalue and singular value methods.

[40] P. Zhlobich. Differential qd algorithm with shifts for rank-structured matrices. *SIAM J. Matrix Anal. Appl.*, 33(4):1153–1171, 2012.