

 Open access • Journal Article • DOI:10.1137/11085219X

A Generalization of the Multishift QR Algorithm — [Source link](#)





Raf Vandebril, David S. Watkins

Published on: 26 Jul 2012 - SIAM Journal on Matrix Analysis and Applications (Society for Industrial and Applied Mathematics)

Topics: QR algorithm, Hessenberg matrix and Eigenvalues and eigenvectors

Related papers:

- [Chasing Bulges or Rotations? A Metamorphosis of the QR-Algorithm](#)
- [Computing approximate extended Krylov subspaces without explicit inversion](#)
- [Fast Computation of the Zeros of a Polynomial via Factorization of the Companion Matrix](#)
- [Eigenvalue and singular value methods](#)
- [A Fast QR Algorithm for Companion Matrices](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/a-generalization-of-the-multishift-qr-algorithm-3f1n94d9oc>

A Generalization of the Multishift QR Algorithm

Raf Vandebril and David S. Watkins

Raf Vandebril
Department of Computer Science
KU Leuven, Belgium
Raf.Vandebril@cs.kuleuven.be

David S. Watkins
Department of Mathematics
Washington State University, USA
Watkins@math.wsu.edu

Abstract

Recently a generalization of Francis's implicitly shifted QR algorithm was proposed, notably widening the class of matrices admitting low-cost implicit QR steps. This unifying framework covered the methods and theory for Hessenberg and inverse Hessenberg matrices and furnished also new, single-shifted, QR-type methods for, e.g., CMV matrices. Convergence of this approach was only suggested by numerical experiments. No theoretical claims supporting the results were presented. In this paper we present multishift variants of these new algorithms. We also provide a convergence theory that shows that the new algorithm performs nested subspace iterations on rational Krylov subspaces. Numerical experiments confirm the validity of the theory.

Article information

- Vandebril, Raf; Watkins, David S., *A generalization of the multishift QR-algorithm*, SIAM Journal on Matrix Analysis and Applications, volume 33, issue 3, pages 759-779, 2012
- This article equals the final publisher's version. A link is found below.
- Journal's homepage: <https://www.siam.org/journals/simax.php>
- Published version: <http://dx.doi.org/10.1137/11085219X>
- KU Leuven's repository url: <https://lirias.kuleuven.be/handle/123456789/365639>

A GENERALIZATION OF THE MULTISHIFT QR ALGORITHM*

RAF VANDEBRIL[†] AND DAVID S. WATKINS[‡]

Abstract. Recently a generalization of Francis’s implicitly shifted QR algorithm was proposed, notably widening the class of matrices admitting low-cost implicit QR steps. This unifying framework covered the methods and theory for Hessenberg and inverse Hessenberg matrices and furnished also new, single-shifted, QR-type methods for, e.g., CMV matrices. Convergence of this approach was only suggested by numerical experiments. No theoretical claims supporting the results were presented. In this paper we present multishift variants of these new algorithms. We also provide a convergence theory that shows that the new algorithm performs nested subspace iterations on rational Krylov subspaces. Numerical experiments confirm the validity of the theory.

Key words. CMV matrices, eigenvalues, QR algorithm, rational Krylov, subspace iteration

AMS subject classifications. 65F15, 15A18

DOI. 10.1137/11085219X

1. Introduction. Francis’s implicitly shifted QR algorithm [7] continues to be the most important method for computing eigenvalues and eigenvectors of matrices. Before we can apply Francis’s algorithm to a given matrix, we must transform the matrix to a condensed form, usually upper Hessenberg form. In [13] a large family of condensed forms was introduced, of which upper Hessenberg form is just one special case. For each of those condensed forms, both an explicit and implicit QR-like algorithm were introduced. Numerical experiments showing good results were presented, but no convergence theory was given. In the current work we introduce multishift algorithms, which perform multiple steps of any degree. In particular, this allows double steps with complex conjugate shifts in real arithmetic. We also supply a convergence theory, and we present results of numerical experiments that support the theory. The MATLAB code is available for the interested reader at <http://people.cs.kuleuven.be/~raf.vandebril/>.

Our condensed matrices are always stored in a factored form

$$A = G_{i_1} G_{i_2} \cdots G_{i_{n-1}} R,$$

where R is upper triangular and each G_{i_j} is a unitary “core” transformation that acts on only two consecutive rows. When A is unitary, R must also be unitary and can be taken to be the identity matrix. In this case the algorithms described in this paper reduce to fast algorithms for the unitary eigenvalue problem, capable of computing the complete set of eigenvalues in $O(n^2)$ flops. These are related to algorithms that have been proposed in the past. If we order the G_i so that A is in upper Hessenberg form, our algorithm is a multishift variant of the unitary QR algorithm

*Received by the editors October 18, 2011; accepted for publication (in revised form) by J. L. Barlow March 26, 2012; published electronically July 26, 2012. This research was partially supported by the Research Council KU Leuven: OT/11/055 Spectral Properties of Perturbed Normal Matrices and their Applications.

<http://www.siam.org/journals/simax/33-3/85219.html>

[†]Department of Computer Science, KU Leuven, 3001 Leuven (Heverlee), Belgium (raf.vandebril@cs.kuleuven.be). This author’s work was supported by the grant “Postdoctoraal Onderzoeker” from the Fund for Scientific Research–Flanders (Belgium).

[‡]Department of Mathematics, Washington State University, Pullman, WA 99164-3113 (watkins@math.wsu.edu).

first proposed by Gragg [8]. If we order the core transformations so that A is in CMV form [15, 3, 11], our algorithm is similar to the one proposed by Bunse–Gerstner and Elsner [2]. A different approach to multishift QR for the unitary eigenvalue problems was presented in [4]. Interesting to note is that [12] presents a cache friendly algorithm for the application of multiple sets of rotations thereby elevating performance to level-3 BLAS status, resulting in a high-performance version of the QR algorithm.

Concerning notation, we have the following conventions. Matrices are typeset in an uppercase font: A , Q ; vectors appear as boldface, lowercase letters: \mathbf{p} , \mathbf{v} ; scalars are depicted as lowercase letters, e.g., the elements of matrices and vectors: $A = [a_{ij}]_{ij}$ and $\mathbf{p} = [p_j]_j$; uppercase calligraphic letters stand for subspaces: \mathcal{K} , \mathcal{E} .

2. Preliminaries. Let A be an $n \times n$ complex matrix whose eigenvalues we would like to compute. We must assume A is nonsingular; the inverse matrix will play a big role in the development. We will not store A in the conventional way, but as a product of core transformations and an upper-triangular matrix. A *core transformation* is a nonsingular matrix G_i that agrees with the identity matrix except in the 2×2 submatrix at the intersection of rows i and $i + 1$ and columns i and $i + 1$. We say that G_i acts on rows i and $i + 1$ because the operation $X \rightarrow G_i X$ modifies rows i and $i + 1$ of X , leaving the other rows untouched. The inverse of a core transformation is a core transformation. Although one can consider nonunitary core transformations, we will restrict our attention to core transformations that are unitary in this paper. Therefore, whenever we speak of a core transformation, we will mean *unitary* core transformation. Unless otherwise stated, we will follow the convention that the subscript i on the core transformation G_i indicates that it acts on rows i and $i + 1$.

The matrix A is said to be of *condensed form* if it admits a QR decomposition

$$(2.1) \quad A = G_{i_1} \cdots G_{i_{n-1}} R,$$

where i_1, \dots, i_{n-1} is a permutation of the integers $1, \dots, n - 1$, each G_{i_j} is a core transformation, and R is upper triangular. Not every permutation leads to a distinct condensed form. If the indices i and j differ by more than one, we have $G_i G_j = G_j G_i$, so it does not matter whether j goes before or after i . However, since G_i does not generally commute with G_{i+1} , it does matter whether G_i is to the left or to the right of G_{i+1} in the factorization. This information is recorded in a *position vector* \mathbf{p} associated with A . This $(n - 2)$ -tuple takes $p_i = \ell$ if G_i is to the left of G_{i+1} , and $p_i = r$ if G_i is to the right of G_{i+1} .

In [13] it was shown that every complex $n \times n$ matrix is unitarily similar to a matrix in condensed form (2.1). The position vector \mathbf{p} can be chosen arbitrarily. The similarity transformation can be effected by a direct method with $O(n^3)$ complexity.

The condensed form (2.1) is a QR factorization of A with the Q factor further decomposed into core transformations. We will call this a *detailed factorization*.

A core transformation G_i will be called *nontrivial* if it is not upper triangular, i.e., $g_{i+1,i} \neq 0$. Since we are restricting our attention to unitary core transformations here, nontrivial means nondiagonal. A matrix in condensed form (2.1) will be called *irreducible* if each of the core transformations in the factorization is nontrivial. Notice that if any one of the core transformations, say G_j , is trivial, then A has the form

$$A = \begin{bmatrix} H_1 & \\ & H_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

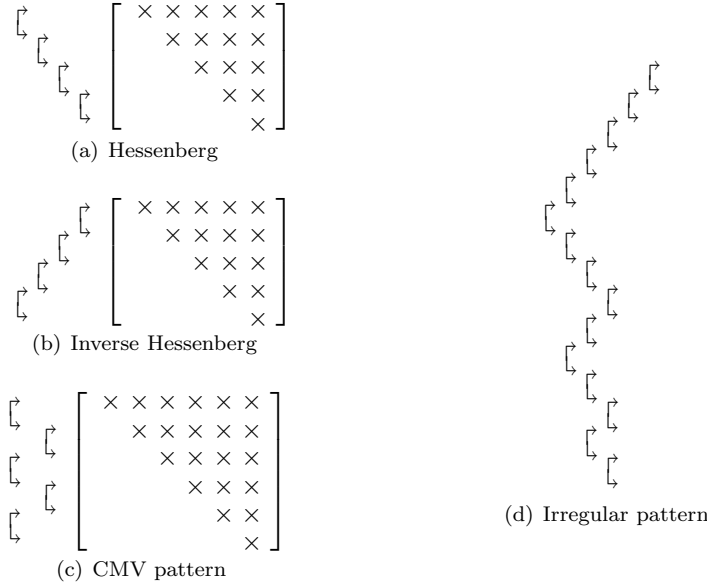


FIG. 2.1. Graphical representation of some factorizations.

where H_1 and H_2 are products of $j-1$ and $n-j-1$ core transformations, respectively. Thus the eigenvalue problem for A reduces to smaller eigenvalue problems for A_{11} and A_{22} , each of which is presented in condensed form.

2.1. Examples. As the ordering of core transformations in a detailed factorization is crucial, we introduce a graphical manner of representing the relative positions of the transformations to aid understanding. Each core transformation is represented by a bracket, with arrows pointing to the rows on which the transformation acts.

Example 2.1 (Hessenberg matrix). The Q factor in the QR factorization of a Hessenberg matrix admits a factorization $Q = G_1 G_2 \cdots G_{n-1}$, often referred to as the Schur parameterization [9]. The position vector is $\mathbf{p} = [\ell, \ell, \dots, \ell]$. Figure 2.1(a) shows the factorization, with a *descending* sequence of core transformations.

Example 2.2 (inverse Hessenberg). The Q factor of a QR factorization of an inverse Hessenberg matrix is of lower Hessenberg form, thus admitting a factorization $Q = G_{n-1} G_{n-2} \cdots G_1$. The position vector $\mathbf{p} = [r, r, \dots, r]$ corresponds to an *ascending* sequence of core transformations as shown in Figure 2.1(b).

Example 2.3 (CMV pattern). A unitary matrix Q with a factorization (consider n even) $Q = G_1 G_3 \cdots G_{n-1} \cdot G_2 G_4 \cdots G_{n-2}$ and associated position vector $[\ell, r, \ell, r, \ell, \dots]$ is called a CMV matrix [15, 10, 3, 11]. In Figure 2.1(c) the alternating pattern, called a *zigzag* pattern, is visualized.

Example 2.4 (irregular pattern). In Figure 2.1(d) an irregular pattern is shown, and the upper-triangular matrix is omitted. The position vector is of the form $\mathbf{p} = [r, r, r, r, r, \ell, \ell, \ell, \ell, r, \ell, \ell, \ell, r]$.

These generalizations of upper Hessenberg form have been studied recently in other contexts. In particular, Fiedler matrices [5, 6] are of this form. See also the recent work [1]. The possibility of such generalizations was first mentioned by Kimura [10], who was also the first to mention CMV matrices, as far as we know.

3. Rational Krylov spaces associated with a condensed form. It was shown in [13] that the unitary similarity transformation that maps a matrix to a given condensed form (2.1) is essentially uniquely determined. The argument depends on a connection with a certain sequence of rational Krylov subspaces associated with the condensed form. This connection is also crucial to the description of the generalizations of Francis's algorithm that we are going to introduce, and it is needed for the convergence theory of the algorithms.

Let \mathbf{p} be the position vector associated with a given condensed form (2.1). For $k = 1, 2, 3, \dots, n-1$, we define the rational Krylov space $\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v})$ to be a space spanned by k vectors from the bilateral sequence

$$(3.1) \quad \dots, A^3\mathbf{v}, A^2\mathbf{v}, A\mathbf{v}, \mathbf{v}, A^{-1}\mathbf{v}, A^{-2}\mathbf{v}, A^{-3}\mathbf{v}, \dots,$$

where the vectors are taken in an order determined by the position vector \mathbf{p} . The first vector in the sequence is \mathbf{v} . The second vector is taken to be either $A\mathbf{v}$ or $A^{-1}\mathbf{v}$, depending upon whether $p_1 = \ell$ or r , respectively. In general, once i vectors have been chosen from the sequence (3.1), the next vector is taken to be the first vector to the left of \mathbf{v} that has not already been taken if $p_i = \ell$. Otherwise ($p_i = r$), we choose the first vector to the right of \mathbf{v} that has not already been taken.

Example 3.1 (Hessenberg matrix). For a Hessenberg matrix the associated position vector comprises only ℓ 's. It follows that the standard Krylov subspaces are retrieved:

$$\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) = \text{span}\{\mathbf{v}, A\mathbf{v}, \dots, A^{k-1}\mathbf{v}\} = \mathcal{K}_k(A, \mathbf{v}).$$

Example 3.2 (inverse Hessenberg matrix). An inverse Hessenberg matrix is characterized by the exclusive appearance of r 's in the associated position vector. We acquire the standard Krylov subspaces built from the inverse of A :

$$\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) = \text{span}\{\mathbf{v}, A^{-1}\mathbf{v}, \dots, A^{-k+1}\mathbf{v}\} = \mathcal{K}_k(A^{-1}, \mathbf{v}).$$

Example 3.3 (CMV pattern). The alternating occurrence of ℓ and r in the position vector $[\ell, r, \ell, r, \ell, \dots]$ linked to a CMV matrix leads to

$$\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) = \text{span}\{\mathbf{v}, A\mathbf{v}, A^{-1}\mathbf{v}, A^2\mathbf{v}, A^{-2}\mathbf{v}, \dots\}.$$

Example 3.4 (irregular pattern). The irregular pattern depicted in Figure 2.1(d) corresponds to the position vector $\mathbf{p} = [r, r, r, r, r, \ell, \ell, \ell, r, r, \ell, \ell, r, \ell, r]$, which yields

$$\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) = \text{span}\{\mathbf{v}, A^{-1}\mathbf{v}, A^{-2}\mathbf{v}, A^{-3}\mathbf{v}, A^{-4}\mathbf{v}, A^{-5}\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, A^3\mathbf{v}, A^{-6}\mathbf{v}, \dots\}.$$

Each rational Krylov space $\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v})$ is just a power shifted version of a standard Krylov space associated with A or A^{-1} .

LEMMA 3.5. Suppose that in the first $k-1$ components of \mathbf{p} the symbol ℓ appears i times and the symbol r appears j times ($i+j = k-1$). Then

$$\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) = \text{span}\{A^{-j}\mathbf{v}, \dots, A^i\mathbf{v}\} = A^{-j}\mathcal{K}_k(A, \mathbf{v}) = A^i\mathcal{K}_k(A^{-1}, \mathbf{v}).$$

Standard Krylov spaces obey the inclusion $A\mathcal{K}_k(A, \mathbf{v}) \subset \mathcal{K}_{k+1}(A, \mathbf{v})$. Here we must be a bit more careful.

LEMMA 3.6. For $k = 1, \dots, n-2$,

- (a) if $p_k = \ell$, then $A\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) \subseteq \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{v})$.
- (b) if $p_k = r$, then $A^{-1}\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) \subseteq \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{v})$.

Proof. Suppose $\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v}) = \text{span}\{A^{-j}\mathbf{v}, \dots, A^i\mathbf{v}\}$. If $p_k = \ell$, then $\mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{v}) = \text{span}\{A^{-j}\mathbf{v}, \dots, A^{i+1}\mathbf{v}\}$, while if $p_k = r$, then $\mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{v}) = \text{span}\{A^{-j-1}\mathbf{v}, \dots, A^i\mathbf{v}\}$. \square

For $k = 1, \dots, n$, define $\mathcal{E}_k = \text{span}\{\mathbf{e}_1, \dots, \mathbf{e}_k\}$, with \mathbf{e}_j the j th canonical basis vector. For each k , \mathcal{E}_k is invariant under the upper-triangular matrix R and under all core transformations G_j except G_k . If $\mathbf{x} \in \mathcal{E}_k$ and $G_j\mathbf{x} = \mathbf{y}$, then $y_k = x_k$ unless j is $k-1$ or k .

THEOREM 3.7. *Let A be a matrix in the irreducible condensed form (2.1) with associated position vector \mathbf{p} . Then for $k = 1, \dots, n-1$,*

$$\mathcal{E}_k = \mathcal{K}_{\mathbf{p},k}(A, \mathbf{e}_1).$$

Proof. The proof is by induction on k . The case $k = 1$ is trivial. Now let us show that if $\mathcal{E}_k = \mathcal{K}_{\mathbf{p},k}(A, \mathbf{e}_1)$, then $\mathcal{E}_{k+1} = \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$. Since $\mathcal{K}_{\mathbf{p},k}(A, \mathbf{e}_1) \subseteq \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$, we automatically have $\mathcal{E}_k \subseteq \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$. If we can show that $\mathbf{e}_{k+1} \in \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$, we will be done.

We can rewrite (2.1) as

$$A = G_L G_k G_R R,$$

where G_L and G_R are the products of the factors to the left and right of G_k , respectively. (This factorization is not uniquely determined; any such factorization can be used.) We consider two cases.

First suppose $p_k = \ell$. Then, using part (a) of Lemma 3.6 with $\mathbf{x} = \mathbf{e}_1$, we have $A\mathcal{E}_k \subseteq \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$. Using the above factorization, we are going to show the existence of $\mathbf{x} \in \mathcal{E}_k$ and $\mathbf{z} \in \mathcal{E}_{k+1}$ with $z_{k+1} \neq 0$, such that $\mathbf{z} = A\mathbf{x}$. If we can do this, then $\mathbf{z} \in A\mathcal{E}_k$, so $\mathbf{z} \in \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$. By the form of \mathbf{z} , and since $\mathbf{e}_1, \dots, \mathbf{e}_k \in \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$, we easily deduce that $\mathbf{e}_{k+1} \in \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$.

It remains to produce vectors \mathbf{x} and \mathbf{z} such that $\mathbf{z} = A\mathbf{x}$, $\mathbf{x} \in \mathcal{E}_k$, $\mathbf{z} \in \mathcal{E}_{k+1}$, and $z_{k+1} \neq 0$. Since G_R does not contain the factor G_k , the space \mathcal{E}_k is invariant under $G_R R$. Since this matrix is nonsingular, it maps \mathcal{E}_k onto \mathcal{E}_k . Thus there is an $\mathbf{x} \in \mathcal{E}_k$ such that $G_R R\mathbf{x} = \mathbf{e}_k$. Let $\mathbf{y} = G_k \mathbf{e}_k$. Then $\mathbf{y} \in \mathcal{E}_{k+1}$ and, because G_k is nontrivial, $y_{k+1} \neq 0$. Let $\mathbf{z} = G_L \mathbf{y}$. Since $p_k = \ell$, G_L does not contain the factor G_{k+1} . Thus \mathcal{E}_{k+1} is invariant under G_L , so $\mathbf{z} \in \mathcal{E}_{k+1}$. Moreover $z_{k+1} = y_{k+1} \neq 0$ because G_L does not contain the factor G_k . Putting the pieces together, we have $\mathbf{z} = A\mathbf{x}$, where $\mathbf{x} \in \mathcal{E}_k$, $\mathbf{z} \in \mathcal{E}_{k+1}$, and $z_{k+1} \neq 0$.

Now we consider the case $p_k = r$. Using part (b) of Lemma 3.6, we have $A^{-1}\mathcal{E}_k \subseteq \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$. If we can show the existence of $\mathbf{x} \in \mathcal{E}_k$, $\mathbf{z} \in \mathcal{E}_{k+1}$ with $\mathbf{z} = A^{-1}\mathbf{x}$ and $z_{k+1} \neq 0$, we will have $\mathbf{z} \in \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$, and we can deduce as in the previous case that $\mathbf{e}_{k+1} \in \mathcal{K}_{\mathbf{p},k+1}(A, \mathbf{e}_1)$.

To produce \mathbf{x} and \mathbf{z} with the desired properties, we use the factorization

$$A^{-1} = R^{-1} G_R^{-1} G_k^{-1} G_L^{-1}$$

and make a similar argument as in the previous case. It is crucial that, since $p_k = r$, G_{k+1}^{-1} is a factor of G_L^{-1} , not of G_R^{-1} . \square

4. Juggling core transformations. Our generalization of the implicitly shifted QR algorithm will be described in terms of three types of operations on core transformations, which we call *passing through*, *fusion*, and *translation*.

In the *passing through* operation, core transformations are “passed through” upper-triangular matrices. Consider a product $G_i R$, where G_i is a core transformation acting on rows i and $i + 1$. If we multiply the factors G_i and R together, we get a product that is upper triangular, except for a bulge in position $(i + 1, i)$. The bulge can be removed by applying a core transformation acting on the right on columns i and $i + 1$, resulting in a new factorization $\tilde{R} \tilde{G}_i$, where \tilde{R} is upper triangular. We have $G_i R = \tilde{R} \tilde{G}_i$, so the core transformation has been passed from the left to the right of the upper-triangular matrix. In a similar manner we can pass a core transformation from right to left. Now if we have a long sequence of core transformations in a particular pattern to the left of an upper-triangular matrix, we can pass the core transformations through one by one so that the same pattern of core transformations emerges on the right-hand side, e.g.,

$$\begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} \left[\begin{array}{ccccc} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{array} \right] = \left[\begin{array}{ccccc} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{array} \right] \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array}.$$

The triangular matrices are not equal, and neither are the core transformations on the left equal to those on the right. What is preserved is the *pattern* of the core transformations.

If two unitary core transformations act on the same two rows, then their product is a unitary core transformation. The act of combining two core transformations in this way is called *fusion* and is depicted graphically as

$$\begin{array}{c} \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \end{array} = \begin{array}{c} \updownarrow \\ \updownarrow \end{array}.$$

The only other operation we need to describe is the *translation* operation, which depends on the *turn-over* (or *shift-through*) operation, depicted by

$$\begin{array}{c} \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \end{array} = \begin{array}{c} \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \end{array}.$$

Here we have three core transformations, one acting on rows i and $i + 1$ sandwiched between two others acting on rows $i - 1$ and i . The product of these can be rewritten as a different product of three core transformations, one acting on rows $i - 1$ and i sandwiched between two acting on rows i and $i + 1$. The proof that this can be done is based on two variants for factoring a 3×3 unitary matrix [14].

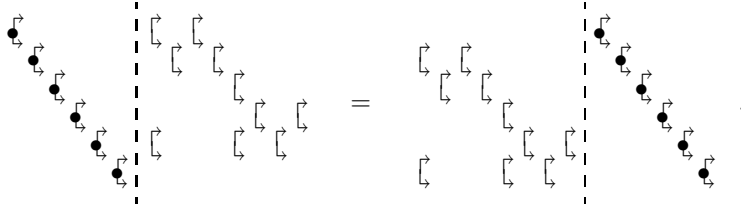
Having a descending or ascending sequence of sufficient length, a *translation* operation can be carried out. Consider first a very simple case:

$$(4.1) \quad \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} = \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} = \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array} = \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \\ \updownarrow \end{array}.$$

On the far left-hand side of (4.1) there is a descending sequence of core transformations with one additional core transformation to the right. This lone transformation is brought into contact with two of the transformations of the descending sequence, a

turn-over transformation is done, a new descending sequence is formed, and a lone core transformation now sits to the left of the descending sequence. We started with a spare core transformation on the right and we ended with a spare transformation on the left. The spare transformation also moved down one position in the process. This is important.

Now consider the more complicated situation:



The descending sequence has several core transformations to the right of it. These can be passed through the descending sequence one at a time, as described above, resulting in the same pattern emerging to the left of the descending sequence, except that the pattern has been moved down by one row. This is what we call a *translation* through a descending sequence.

Translations through ascending sequences are defined similarly. The picture is just the mirror image of that of a translation through a descending sequence. Core transformations can be passed from left to right through an ascending sequence with the pattern of the core transformations moving down by one row.

5. Generalization of Francis's implicitly shifted QR iteration. Generalizations of both the explicit and implicit QR algorithms were given in [13]. There it was shown that if the matrix A has condensed form with position vector \mathbf{p} , then the matrix \hat{A} resulting from one iteration has a different position vector $\hat{\mathbf{p}}$ derived from \mathbf{p} by moving each entry up by one. The vacated bottom entry can be filled in with either an ℓ or an r , and we can control which one it is. For example, if $\mathbf{p} = [\ell, r, r, r, \ell]$, then $\hat{\mathbf{p}}$ will be either $[r, r, r, \ell, \ell]$ or $[r, r, r, \ell, r]$, whichever we prefer.

We are now going to introduce a multishift algorithm that does steps of degree m , i.e., m single steps all at once. Based on our experience with single steps, we would expect that after such a multiple step, the entries of the position vector would be moved upward by m positions. This is indeed what happens. For example, if we have $\mathbf{p} = [\ell, r, r, r, r, \ell, \ell, \ell]$ and we then do a step of degree $m = 3$, we obtain a matrix whose condensed form has the position vector $\hat{\mathbf{p}} = [r, r, \ell, \ell, \ell, a, b, c]$, where $[a, b, c]$ can be made to be any of the eight possible combinations of the two symbols ℓ and r .

An iteration of degree m begins with the choice of m nonzero shifts ρ_1, \dots, ρ_m . Suppose that in the first m positions of \mathbf{p} , the symbols ℓ and r appear i and j times, respectively. The iteration begins by calculating a vector

$$(5.1) \quad \mathbf{x} = \alpha A^{-j} (A - \rho_1 I) \cdots (A - \rho_m I) \mathbf{e}_1.$$

The scalar α is any convenient scaling constant. Letting

$$p(z) = (z - \rho_1) \cdots (z - \rho_m),$$

we can write this as

$$\mathbf{x} = \alpha A^{-j} p(A) \mathbf{e}_1.$$

The equation

$$A^{-1}(A - \rho I) = -\rho(A^{-1} - \rho^{-1}I)$$

shows that \mathbf{x} can also be expressed as

$$(5.2) \quad \mathbf{x} = \beta A^i (A^{-1} - \rho_1^{-1}I) \cdots (A^{-1} - \rho_m^{-1}I) \mathbf{e}_1$$

or, letting $q(z) = (z - \rho_1^{-1}) \cdots (z - \rho_m^{-1})$,

$$\mathbf{x} = \beta A^i q(A^{-1}) \mathbf{e}_1.$$

Alternatively we can express \mathbf{x} using a product containing i factors of the form $A - \rho I$ and j factors of the form $A^{-1} - \rho^{-1}I$, in any order, with no additional power of A . The following algorithm gives a procedure for computing \mathbf{x} :

$$(5.3) \quad \begin{array}{l} \mathbf{x} \leftarrow \mathbf{e}_1 \\ \text{for } i = 1, \dots, m, \\ \quad \left[\begin{array}{l} \text{if } p_i = \ell \\ \quad \mathbf{x} \leftarrow \alpha_i (A - \rho_i I) \mathbf{x}, \\ \text{if } p_i = r \\ \quad \mathbf{x} \leftarrow \alpha_i (I - \rho_i A^{-1}) \mathbf{x}. \end{array} \right. \end{array}$$

The α_i are any convenient scaling factors.

THEOREM 5.1. *The vector \mathbf{x} given by (5.1), (5.2), or (5.3) satisfies $\mathbf{x} \in \mathcal{E}_{m+1}$.*

Proof. Let $\mathbf{x}^{(i)}$ denote the vector \mathbf{x} after i steps of (5.3). Using Lemma 3.6 and Theorem 3.7 we see by induction on i that $\mathbf{x}^{(i)} \in \mathcal{E}_{i+1}$. \square

Once \mathbf{x} has been computed, one can build m core transformations S_1, \dots, S_m such that

$$(5.4) \quad S_1^{-1} \cdots S_m^{-1} \mathbf{x} = \gamma \mathbf{e}_1.$$

Here, we continue with the convention that S_i acts on rows i and $i + 1$. Next we transform A to

$$(5.5) \quad S_1^{-1} \cdots S_m^{-1} A S_m \cdots S_1.$$

The rest of the iteration will consist of returning this matrix to a condensed form. To get an idea what this matrix looks like, consider a simple case. Suppose A is 4×4 with $\mathbf{p} = [\ell, \ell]$ (upper Hessenberg):

$$A = \begin{array}{c} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix} \end{array}.$$

$S_1^{-1} \cdots S_m^{-1}$ and $S_m \cdots S_1$ are descending and ascending sequences of core transformations, respectively, so in the case $m = 2$, for example,

$$S_1^{-1} S_2^{-1} A S_2 S_1 = \begin{array}{c} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix} \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array} \end{array}.$$

$$S_1^{-1}S_2^{-1}AS_2S_1 = \begin{array}{c} \begin{array}{c} \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \\ \updownarrow \end{array} \begin{array}{c} \updownarrow \\ \updownarrow \end{array} \left[\begin{array}{cccc} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{array} \right].$$

(5.6) 

- If $p_{m-1} = \ell$, translate limb to left through descending branch. Otherwise, translate limb to right through ascending branch.
- If $p_m = \ell$, fuse m th core transformation in main branch with m th core transformation of descending branch. Otherwise, fuse with ascending branch.

¹This is exactly the correct picture in the case when A is unitary.

with the descending sequence. After Step 0 we have

$$(5.7) \quad \begin{array}{c} \begin{array}{c} \bullet \\ \downarrow \\ \bullet \\ \downarrow \\ \bullet \end{array} \begin{array}{c} \leftarrow \\ \leftarrow \\ \leftarrow \end{array} \begin{array}{c} \rightarrow \\ \rightarrow \\ \rightarrow \end{array} \end{array} .$$

These same operations are shown schematically for a larger example in Figure 5.1. This matrix has dimension 22, so there are 21 core transformations, indicated by dots in the first picture. The position vector is $\mathbf{p} = [r, \ell, \ell, \ell, \ell, r, \ell, r, \ell, \ell, \ell, r, r, r, r, \ell, \ell, r, r, r]$. The degree of the step is $m = 4$.

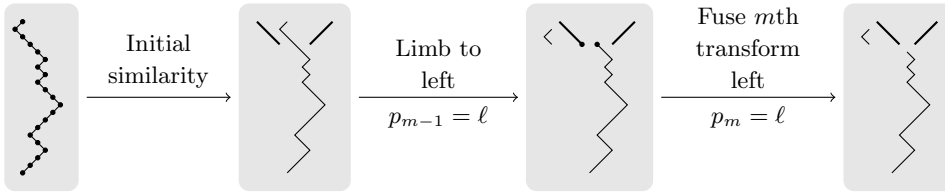


FIG. 5.1. Step 0: preparatory step.

Main loop. Once Step 0 is complete, we proceed to steps $k = 1, 2, \dots, n - 2$. Immediately before step k , the matrix consists of a finished part comprised of $k - 1$ core transformations (if $k > 1$), followed by descending and ascending branches of length m , followed by a not-yet-touched part of length $n - m - k$ (if $k < n - m$). There is also a limb of length $m - 1$, which lies either to the left of the descending branch or to the right of the ascending branch.

If $p_{m+k} = \ell$, the descending branch plus the top two transformations of the not-yet-touched part form a descending sequence of length $m + 2$. Transformation $m + 1$ becomes a part of the descending branch. The ascending branch will be translated through the descending branch, moving it downward. Transformation $m + 2$ remains untouched by this step, but it is in the right position not to interfere with the translation. If $p_{m+k} = r$, the roles of the descending and ascending branches are reversed. The value of p_{m+k} determines the value of \hat{p}_k . The complete step k ($k < n - m - 1$) is as follows:

- Set $\hat{p}_k = p_{m+k}$.
- If $\hat{p}_k = \ell$, augment descending branch by adjoining first transformation from not-yet-touched part. Translate ascending branch to left through descending branch.
- If $\hat{p}_k = r$, augment ascending branch and translate descending branch to right through descending branch.
- If limb is on left, translate limb to right through ascending branch. Otherwise, translate limb to left through descending branch. Limb is now in middle.
- If $\hat{p}_k = \ell$, perform unitary similarity transformation that eliminates ascending branch from the left and makes it appear on the right. Pass ascending branch through upper-triangular matrix. Transfer top core transformation from descending branch to finished part. Position of this transformation is consistent with the choice $\hat{p}_k = \ell$. Limb ends up on left.
- If $\hat{p}_k = r$, pass ascending branch to right, through upper-triangular matrix. Then perform unitary similarity transformation that eliminates descending

In (5.7) we have $p_{m+k} = p_4 = r$ (with $m = 3$ and $k = 1$), so we set $\hat{p}_1 = r$ and translate the descending sequence through the ascending sequence. Then, since the limb is on the left, we translate it to the right through the ascending sequence to obtain

(5.8) 

(5.9)

For our larger 22×22 example, step 1 is shown in Figure 5.2. The first core transformation of the final pattern is marked by a dot in the final picture. The second step is depicted in Figure 5.3. Subsequent steps, up to step $16 = n - m - 2$, are depicted in Figure 5.4.

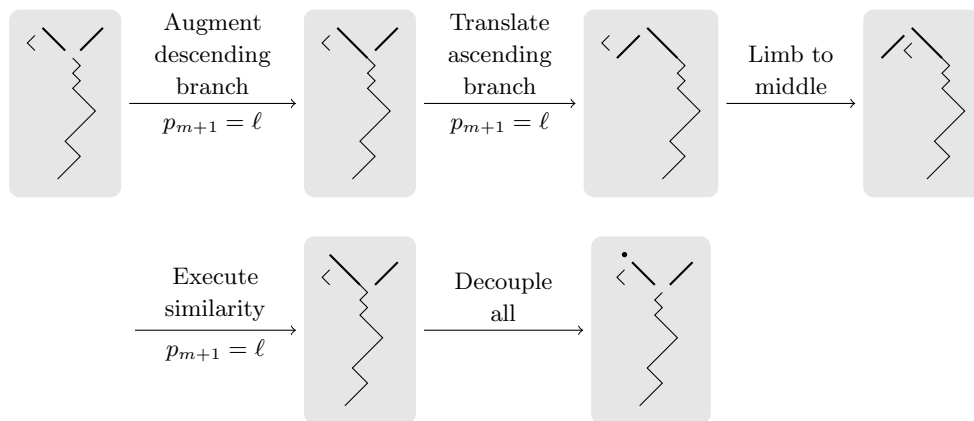


FIG. 5.2. *Main loop for $k = 1$.*

Step $n-1-m$. At the beginning of step $k = n-1-m$, there is only one remaining not-yet-touched core transformation. Now we have a choice. We can set $\hat{p}_k = \ell$ and

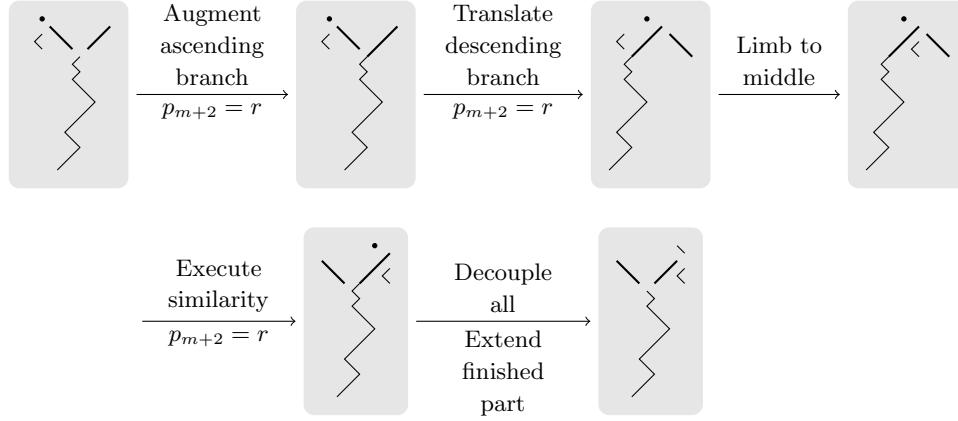
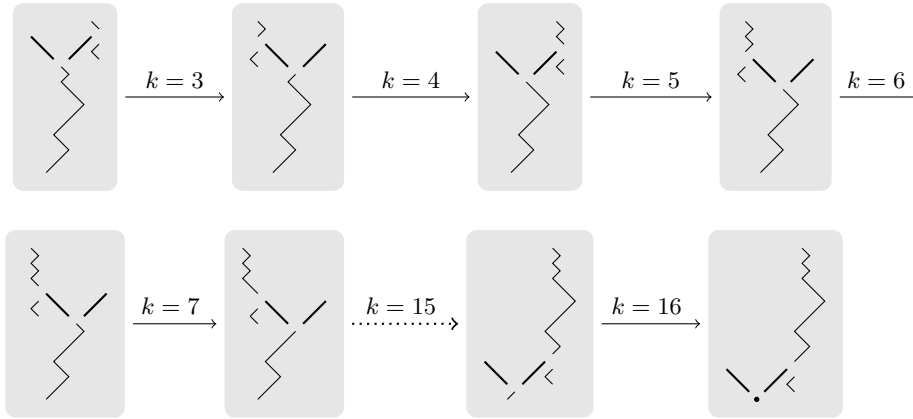
FIG. 5.3. Main loop for $k = 2$.

FIG. 5.4. Several chasing steps.

adjoin the final translation to the descending branch, or we can set $\hat{p}_k = r$ and adjoin the final transformation to the ascending branch. Then we proceed exactly as in the previous steps. Then we do one more thing. Since there is no more untouched part of the matrix, the bottom transformations of the ascending and descending branches are adjacent and can be fused.

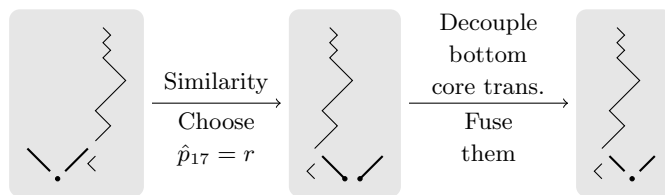
- Remove bottom core transformations from descending and ascending branches and fuse them. The resulting core transformation will be called the *bottom transformation*.

The descending and ascending branches now have length $m - 1$. The limb still has length $m - 1$. Its bottom transformation is now at the same level as the bottom transformation. That is, it is at the bottom of the matrix.

In our small example (5.9), at step $k = 2$, we are already at step $n - m - 1$. We can choose to translate the ascending sequence through the descending sequence or vice versa. Suppose we decide to do the former (choice $\hat{p}_2 = \ell$). We translate the ascending sequence through the descending sequence. Then, since the limb is on the right, we also pass it through the descending sequence. Then we effect a similarity transformation that removes the ascending sequence from the left and places it on the

(5.10) 

For our larger example, step $n - 1 - m = 17$ is shown in Figure 5.5.

FIG. 5.5. *Step $n - 1 - m$.*

- If $\hat{p}_k = \ell$, adjoin bottom transformation to descending branch, then translate ascending branch to left through descending branch.
- If $\hat{p}_k = r$, adjoin bottom transformation to ascending branch, then translate descending branch to right through ascending branch.
- Remove bottom translation from limb. Limb now has length $n - 2 - k$.
- If limb is on left, fuse transformation that was removed from the limb with bottom transformation of ascending branch, if possible. Then translate remaining part of limb to right through ascending branch. If the fusion operation is not possible initially, do the translation first, then fuse.
- If limb is on right, fuse transformation that was removed from the limb with bottom transformation of descending branch, if possible. Then translate remaining part of limb to left through descending branch. If the fusion operation is not possible initially, do the translation first, then fuse.
- If $\hat{p}_k = \ell$, perform unitary similarity transformation that moves ascending branch from left to right. Pass ascending branch through upper-triangular matrix. Transfer top core transformation from descending branch to finished part. Limb ends up on left.
- If $\hat{p}_k = r$, pass descending branch through upper-triangular matrix. Then perform unitary similarity transformation that moves descending branch from right to left. Transfer top core transformation from ascending branch to finished part. Limb ends up on right.

6. Convergence theory. The iteration that we have just described effects a unitary similarity transformation

$$(6.1) \quad \hat{A} = Q^{-1}AQ$$

to a new condensed matrix \hat{A} , which has position vector $\hat{\mathbf{p}}$.

THEOREM 6.1. *Suppose $\hat{A} = Q^{-1}AQ$, where \hat{A} is in irreducible condensed form with position vector $\hat{\mathbf{p}}$. Let $\mathbf{q}_1, \dots, \mathbf{q}_n$ denote the columns of Q , and let $\mathcal{Q}_k = \text{span}\{\mathbf{q}_1, \dots, \mathbf{q}_k\}$, $k = 1, \dots, n$. Then*

$$\mathcal{Q}_k = \mathcal{K}_{\hat{\mathbf{p}},k}(\hat{A}, \mathbf{q}_1), \quad k = 1, \dots, n-1.$$

Proof. Applying Theorem 3.7 to \hat{A} , we have, for $k = 1, \dots, n-1$,

$$\mathcal{E}_k = \mathcal{K}_{\hat{\mathbf{p}},k}(\hat{A}, \mathbf{e}_1).$$

Thus

$$\mathcal{Q}_k = Q\mathcal{E}_k = \mathcal{K}_{\hat{\mathbf{p}},k}(Q\hat{A}Q^{-1}, Q\mathbf{e}_1) = \mathcal{K}_{\hat{\mathbf{p}},k}(A, \mathbf{q}_1). \quad \square$$

The transforming matrix in (6.1) has the form

$$Q = S_m \cdots S_1 U,$$

where U is the product of all of the similarity transformations performed during steps $1, \dots, n-1$. One easily checks that none of those transformations touch the first row, so $U\mathbf{e}_1 = \mathbf{e}_1$.

THEOREM 6.2. *Let $p(z) = (z - \rho_1) \cdots (z - \rho_m)$, where ρ_1, \dots, ρ_m are the shifts. Then*

$$Q\mathbf{e}_1 = \delta \mathbf{x} = \eta A^{-j} p(A) \mathbf{e}_1,$$

where δ and η are scalars.

Proof. Since $U\mathbf{e}_1 = \mathbf{e}_1$, we have $Q\mathbf{e}_1 = S_m \cdots S_1 \mathbf{e}_1$. By (5.4), $S_m \cdots S_1 \mathbf{e}_1$ is a multiple of \mathbf{x} , where \mathbf{x} is given by (5.1). \square

The next theorem, which is our main result, shows that each iteration of our algorithm effects nested subspace iterations with a change of coordinate system [18, section 6.2], [19, p. 396] on a sequence of rational Krylov spaces. It follows that, given good choices of shifts, successive iterates will move quickly toward block-triangular form, leading to a deflation.

THEOREM 6.3. *Consider one iteration of degree m , $\hat{A} = Q^{-1}AQ$, where A and \hat{A} are in irreducible condensed form with position vectors \mathbf{p} and $\hat{\mathbf{p}}$, respectively. For a given k ($1 \leq k \leq n-1$), let $i(k)$ and $j(k)$ denote the number of symbols ℓ and r , respectively, in the first $k-1$ positions of \mathbf{p} ($i(k) + j(k) = k-1$). Define $\hat{i}(k)$ and $\hat{j}(k)$ analogously with respect to $\hat{\mathbf{p}}$. Then the iteration $A \rightarrow \hat{A}$ effects one step of subspace iteration driven by*

$$(6.2) \quad A^{\hat{i}(k)-i(k)-j(m+1)} p(A),$$

followed by a change of coordinate system. By this we mean that

$$A^{\hat{i}(k)-i(k)-j(m+1)} p(A) \mathcal{E}_k = \mathcal{Q}_k.$$

The change of coordinate system $\hat{A} = Q^{-1}AQ$ maps \mathcal{Q}_k back to \mathcal{E}_k .

Proof. Using Theorems 3.7 and 6.2, Lemma 3.5, and Theorem 6.1, in that order, we get

$$\begin{aligned}
 A^{\hat{i}(k)-i(k)-j(m+1)}p(A)\mathcal{E}_k &= A^{\hat{i}(k)-i(k)-j(m+1)}p(A)\mathcal{K}_{\mathbf{p},k}(A, \mathbf{e}_1) \\
 &= A^{\hat{i}(k)-i(k)}\mathcal{K}_{\mathbf{p},k}(A, A^{-j(m+1)}p(A)\mathbf{e}_1) \\
 &= A^{\hat{i}(k)-i(k)}\mathcal{K}_{\mathbf{p},k}(A, \mathbf{q}_1) \\
 &= \mathcal{K}_{\hat{\mathbf{p}},k}(A, \mathbf{q}_1) \\
 &= \mathcal{Q}_k. \quad \square
 \end{aligned}$$

To help clarify this result, consider the two extreme cases. In the Hessenberg case ($\hat{\mathbf{p}} = \mathbf{p} = [\ell, \dots, \ell]$), the driving function (6.2) is $p(A) = (A - \rho_1 I) \cdots (A - \rho_m I)$ for all k , in agreement with standard convergence theory. In the inverse Hessenberg case ($\hat{\mathbf{p}} = \mathbf{p} = [r, \dots, r]$), the driving function is $A^{-m}p(A) \sim (A^{-1} - \rho_1^{-1}I) \cdots (A^{-1} - \rho_m^{-1}I)$ for all k .

Now consider s successive iterations. To begin with we will assume that the same shifts are taken on all iterations. If we consider the subspace iterations that effectively take place in the original coordinate system, we have

$$\mathcal{E}_k \rightarrow A^{\tilde{i}}(p(A))^s \mathcal{E}_k,$$

where \tilde{i} is the difference in the number of times the symbol ℓ appears in the first $k-1$ positions of the position vector of the final condensed matrix versus the initial condensed matrix. The power \tilde{i} is bounded: $-n \leq \tilde{i} \leq n$. If the ℓ - r pattern is periodic with period m , \tilde{i} will be zero.

Let $\lambda_1, \dots, \lambda_n$ denote the eigenvalues of A , ordered so that $|p(\lambda_1)| \geq |p(\lambda_2)| \geq \dots \geq |p(\lambda_n)|$. Then the average improvement per step, taken over a sequence of s steps, is given approximately by the ratio

$$(6.3) \quad |(\lambda_{k+1}/\lambda_k)^{\tilde{i}/s} p(\lambda_{k+1})/p(\lambda_k)|.$$

The term that is dependent on \tilde{i} could be significant in the short term, but it will be insignificant in the long run.

If all the shifts are excellent approximations to m eigenvalues of A , then in the case $k = n - m$ we will have $|p(\lambda_{k+1})/p(\lambda_k)| \ll 1$, and there will be rapid convergence for that value of k (meaning G_k goes rapidly to diagonal form).

If we now make the more realistic assumption that the shifts are changed on each step, (6.3) is replaced by

$$|(\lambda_{k+1}/\lambda_k)^{\tilde{i}} \prod_{i=1}^s p_i(\lambda_{k+1})/p_i(\lambda_k)|^{1/s},$$

where p_i is the polynomial determined by the shifts on the i th iteration. With good shift choices, locally quadratic convergence is obtained.

6.1. Choice of shifts. We tried two similar shifting strategies. The conceptually simplest is to compute the eigenvalues of the lower-right $m \times m$ submatrix and use them as the shifts. This normally yields quadratic convergence [20, 17]. In order to implement this strategy, we must form the lower right-hand submatrix by multiplying some of the core transformations into the upper-triangular matrix. Usually this is a

trivial task. For example, in the Hessenberg case, we just have to multiply in the bottom m core transformations in order from bottom to top. The total cost of the shift computation is $O(m^3)$, which is $O(1)$ if m is a small fixed number. However, in the inverse Hessenberg case, we must multiply in all $n - 1$ core transformations, as we must go in order from nearest to furthest from the triangular matrix. Since we need only update the last m columns, the cost is only $O(nm) = O(n)$, which is not too bad, but it is nevertheless more than we are used to paying for shifts. Note that in the CMV case the computation costs $O(1)$. In the random case the cost will also be $O(1)$ with high probability.

Because this strategy is occasionally more expensive than we would like, we tried another strategy in which we just multiply the bottom m core transformations into R . This strategy uses as shifts the eigenvalues of the lower right-hand $m \times m$ submatrix of

$$G_{j_1} \cdots G_{j_m} R,$$

where j_1, \dots, j_m is a permutation of $n - m, \dots, n - 1$ that leaves these core transformations in the same relative order as they are in the detailed factorization of A . If $p_{n-m-1} = \ell$, this submatrix is exactly the same as the lower right-hand submatrix of A , so this strategy is the same as the first strategy. When $p_{n-m-1} = r$, there will be a difference. However, if $G_{n-m} = I$, which is the case when the bottom $m \times m$ submatrix is disconnected from the rest of the matrix, there is again no difference. If G_{n-m} is close to the identity, as we have when we are close to convergence, then the difference will be slight, and the difference in shifts will also be slight. As a consequence, this strategy should also yield quadratic convergence.

We tried both strategies and found that they gave similar results. For the numerical experiments reported below, we used the second shift strategy.

6.2. Convergence in the single-shift case. Now consider the simple case $m = 1$. With good shifts we expect rapid convergence when $k = n - 1$, so let's focus on that case. Theorem 6.3 gives (for $r(A) = A^{i(n-1)-i(n-1)-j(2)}(A - \rho I)$)

$$\mathcal{Q}_{n-1} = r(A)\mathcal{E}_{n-1}.$$

If we assign numerical values $\ell = 1$ and $r = 0$ to the entries of the position vectors, we have $\hat{i}(n-1) - i(n-1) - j(2) = \hat{p}_{n-2} - p_1 - j(2)$. It is easy to check that $p_1 + j(2) = 1$ always, so $\hat{i}(n-1) - i(n-1) - j(2) = \hat{p}_{n-2} - 1$. Thus

$$r(A) = \begin{cases} (A - \rho I) & \text{if } \hat{p}_{n-2} = \ell, \\ (A^{-1} - \rho^{-1} I) & \text{if } \hat{p}_{n-2} = r. \end{cases}$$

The associated ratios of eigenvalues are

$$\frac{|\lambda_n - \rho|}{|\lambda_{n-1} - \rho|} \quad \text{if } \hat{p}_{n-2} = \ell$$

and

$$\frac{|\lambda_n^{-1} - \rho^{-1}|}{|\lambda_{n-1}^{-1} - \rho^{-1}|} \quad \text{if } \hat{p}_{n-2} = r.$$

Since

$$\frac{\lambda_n^{-1} - \rho^{-1}}{\lambda_{n-1}^{-1} - \rho^{-1}} = \left(\frac{\lambda_{n-1}}{\lambda_n} \right) \left(\frac{\lambda_n - \rho}{\lambda_{n-1} - \rho} \right),$$

it should be better to take $\hat{p}_{n-2} = r$ if and only if $|\lambda_{n-1}| < |\lambda_n|$.

Here the eigenvalues are numbered so that $\lambda_n - \rho$ is the smallest (in magnitude) eigenvalue of $A - \rho I$, and $\lambda_{n-1} - \rho$ is the second smallest. It is assumed that $\lambda_n^{-1} - \rho^{-1}$ is the smallest eigenvalue of $A^{-1} - \rho^{-1}I$, and $\lambda_{n-1}^{-1} - \rho^{-1}$ is the second smallest. This is certainly not always true, but it is typically true, especially if ρ approximates λ_n well.

7. Numerical experiments. We ran some tests with the objective of checking the validity of our theoretical results. The codes, which are written in MATLAB, are available at <http://people.cs.kuleuven.be/~raf.vandebril/>. We generated random matrices of dimension 100, 200, ..., 500 and used four different versions of our code to compute the Schur form and the eigenvalues of each matrix. The first version transforms the matrix to upper Hessenberg form (position vector $\mathbf{p} = [\ell, \ell, \dots, \ell]$) and retains that form throughout the multishift iterations. The second version transforms the matrix to the CMV form ($\mathbf{p} = [\ell, r, \ell, r, \dots]$) and retains that form. The third reduces the matrix to a random form (entries of \mathbf{p} chosen at random) and continues to introduce position values ℓ or r at random (equal probability) throughout the multishift iterations. The fourth transforms the matrix to inverse Hessenberg form ($\mathbf{p} = [r, r, \dots, r]$) and maintains that form throughout the multishift iterations. In our test we did iterations of degree $m = 4$. Blocks were deflated when their size was less than $4m$. The criterion for deflation or splitting of the problem was $|G_i(2, 1)| < 10^{-17}$. In preliminary tests this deflation criterion was compared to the conventional deflation criterion based on comparing magnitudes of the subdiagonal elements. The differences were negligible, so we opted for the cheaper criterion.

After each run we compared the eigenvalues computed by our code with those computed by the MATLAB `eig` command and always had good agreement. We checked relative backward errors by computing $\|AU - UR\|_2 / \|A\|_2$, where $A = URU^H$ is the computed Schur decomposition. These are shown in Figure 7.1.

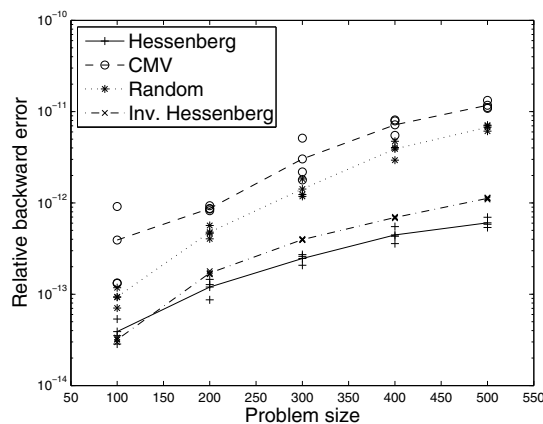


FIG. 7.1. Relative backward error.

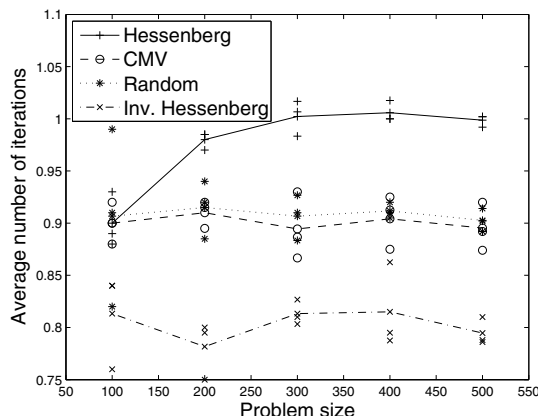


FIG. 7.2. Average iterations per eigenvalue.

Each run was repeated five times. Each backward error is recorded on the graph by an appropriate symbol, and the averages of the five errors are connected by lines. We note that the backward errors are around 10^{-11} or less, and the Hessenberg and inverse Hessenberg errors are a bit better than the others.

Figure 7.2 displays the average number of iterations per eigenvalue, computed by dividing the total number of iterations by the dimension of the matrix. We observe that one or fewer iterations per eigenvalue are required. This is about what one would expect for iterations of degree $m = 4$. It serves as confirmation of our convergence theory in a couple of ways. First of all, we observe convergence. Second, the convergence must be something like quadratic, as otherwise many more iterations would be required. Experience with the quadratically convergent QR algorithm suggests that about 3 to 4 iterations (with $m = 1$) per eigenvalue are needed, and that is what we are observing here.

We also looked for direct evidence of quadratic convergence. According to theory, the entry $|G_{n-m}(2, 1)|$ should converge to zero quadratically. We observed this. However, the actual behavior is more complicated than the theory would suggest. If some (say k) of the shifts approximate eigenvalues much better than the other $(m - k)$ shifts do, it can happen that $|G_{n-k}(2, 1)|$ goes to zero instead. In fact, it often happens that for several different k at once, the numbers $|G_{n-k}(2, 1)|$ are converging to zero. Bearing this in mind, generally quadratic convergence is observed.

Figure 7.2 indicates that the inverse Hessenberg method is some 20% more efficient than the Hessenberg method, with the other two methods in between. This is a consequence of the class of matrices that was used in the experiments: random matrices with independent and identically distributed entries. When we repeated the experiments with the inverses of random matrices, the efficiencies were reversed.

7.1. Shift blurring. The results we have shown for $m = 4$ are typical of what one gets for small values of m less than about 10. It is natural to ask how well the code works for larger values of m such as 20, 30, or 40. Experience (see [16], [17, section 7.1]) suggests that its performance might be diminished due to shift blurring. The simplest outward symptom would be increased iteration counts. We did indeed observe this.

TABLE 7.1
Average number of iterations per deflation.

Number of shifts	10	20	30	40
Hessenberg	3.6	7.2	8.6	8.4
CMV type	3.9	6.2	7.2	7.4
Random	3.7	6.4	7.4	7.7
Inv. Hessenberg	4.1	6.1	8.0	7.5

Table 7.1 shows the average number of iterations per deflation for $m = 10, 20, 30$, and 40 . In these experiments the matrices were 500×500 , and each number is an average over three runs. An iteration of degree m is like m single steps, and a typical deflation delivers about m eigenvalues. It follows that the number of iterations per deflation should be about the same over all values of m . In Table 7.1 we see that for $m = 10$, about four iterations are needed for each deflation. This is similar to what is observed for smaller values of m . But for $m = 20, 30$, and 40 , almost twice as many iterations per deflation are needed. This is symptomatic of shift blurring.

8. Conclusions. We have introduced a family of algorithms that generalizes Francis's implicitly shifted QR algorithm. The new algorithms operate on condensed forms that are generalizations of upper Hessenberg form. Multiple steps of arbitrary degree can be taken. We have also supplied a convergence theory based on rational Krylov subspaces, and we have presented numerical experiments that verify the validity of our theory.

REFERENCES

- [1] R. C. BARROSO AND S. DELVAUX, *Orthogonal Laurent polynomials on the unit circle and snake-shaped matrix factorizations*, J. Approx. Theory, 161 (2009), pp. 65–87.
- [2] A. BUNSE-GERSTNER AND L. ELSNER, *Schur parameter pencils for the solution of the unitary eigenproblem*, Linear Algebra Appl., 154/156 (1991), pp. 741–778.
- [3] M. J. CANTERO, L. MORAL, AND L. VELAZQUEZ, *Five-diagonal matrices and zeros of orthogonal polynomials on the unit circle*, Linear Algebra Appl., 362 (2003), pp. 29–56.
- [4] R. J. A. DAVID AND D. S. WATKINS, *Efficient implementation of the multishift QR algorithm for the unitary eigenvalue problem*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 623–633.
- [5] M. FIEDLER, *A note on companion matrices*, Linear Algebra Appl., 372 (2003), pp. 325–331.
- [6] M. FIEDLER, *Complementary basic matrices*, Linear Algebra Appl., 384 (2004), pp. 199–206.
- [7] J. G. F. FRANCIS, *The QR transformation, part II*, Comput. J., 4 (1961), pp. 332–345.
- [8] W. B. GRAGG, *The QR algorithm for unitary Hessenberg matrices*, J. Comput. Appl. Math., 16 (1986), pp. 1–8.
- [9] W. B. GRAGG AND L. REICHEL, *A divide and conquer method for unitary and orthogonal eigenproblems*, Numer. Math., 57 (1990), pp. 695–718.
- [10] H. KIMURA, *Generalized Schwarz form and lattice-ladder realizations of digital filters*, IEEE Trans. Circuits Systems, 32 (1985), pp. 1130–1139.
- [11] B. SIMON, *CMV matrices: Five years after*, J. Comput. Appl. Math., 208 (2007), pp. 120–154.
- [12] F. G. VAN ZEE, R. A. VAN DE GEIJN, AND G. QUINTANA-ORTÍ, *Restructuring the QR-Algorithm for High-Performance Applications of Givens Rotations*, Technical Report TR-11-36, Department of Computer Science, The University of Texas at Austin, Austin, TX, 2011.
- [13] R. VANDEBRIL, *Chasing bulges or rotations? A metamorphosis of the QR-algorithm*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 217–247.
- [14] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *Matrix Computations and Semiseparable Matrices, Volume I: Linear Systems*, Johns Hopkins University Press, Baltimore, MD, 2008.
- [15] D. S. WATKINS, *Some perspectives on the eigenvalue problem*, SIAM Rev., 35 (1993), pp. 430–471.

- [16] D. S. WATKINS, *The transmission of shifts and shift blurring in the QR algorithm*, Linear Algebra Appl., 241/243 (1996), pp. 877–896.
- [17] D. S. WATKINS, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.
- [18] D. S. WATKINS, *Fundamentals of Matrix Computations*, 3rd ed., John Wiley and Sons, Hoboken, NJ, 2010.
- [19] D. S. WATKINS, *Francis's algorithm*, Amer. Math. Monthly, 118 (2011), pp. 387–403.
- [20] D. S. WATKINS AND L. ELSNER, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.