SIAM Review
Vol. 50, No. 1, pp. 133–145

The *QR* Algorithm Revisited*

David S. Watkins†

Abstract. The QR algorithm is still one of the most important methods for computing eigenvalues and eigenvectors of matrices. Most discussions of the QR algorithm begin with a very basic version and move by steps toward the versions of the algorithm that are actually used. This paper outlines a pedagogical path that leads directly to the implicit multishift QR algorithms that are used in practice, bypassing the basic QR algorithm completely.

Key words. eigenvalue, eigenvector, QR algorithm, Hessenberg form

AMS subject classifications. 65F15, 15A18

DOI. 10.1137/060659454

1. Introduction. For many years the QR algorithm [7], [8], [9], [25] has been our most important tool for computing eigenvalues and eigenvectors of matrices. In recent years other algorithms have arisen [1], [4], [5], [6], [11], [12] that may supplant QR in the arena of symmetric eigenvalue problems. However, for the general, nonsymmetric eigenvalue problem, QR is still king. Moreover, the QR algorithm is a key auxiliary routine used by most algorithms for computing eigenpairs of large sparse matrices, for example, the implicitly restarted Arnoldi process [10], [13], [14] and the Krylov–Schur algorithm [15].

Early in my career I wrote an expository paper entitled "Understanding the QR algorithm," which was published in *SIAM Review* in 1982 [16]. It was not my first publication, but it was my first paper in numerical linear algebra, and it marked a turning point in my career: I've been stuck on eigenvalue problems ever since.

Over the years my view of the QR algorithm has evolved substantially, and I have revisited the question of how to explain it and its variants on several occasions [17], [18], [19], [21], [22], [23]. One might think I have said quite enough by now, but just recently I felt the urge to visit this topic once again.

What bothers me is this. Most discussions of the QR algorithm begin with the equations

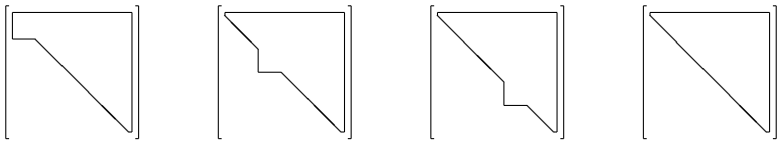
$$A_{k-1} = Q_k R_k, \qquad R_k Q_k = A_k. \tag{1.1}$$

In words, to get from iterate $k-1$ to iterate k, do a QR factorization of A_{k-1}. Then multiply the factors back together in the reverse order to get A_k. This is the basic QR algorithm. It is appealingly simple, but it is also far removed from the versions of the QR algorithm that are used in practice. The latter are bulge-chasing algorithms: Each iteration begins with an upper Hessenberg matrix, makes an initial transformation that produces a bulge in the Hessenberg form at the top of the matrix, then chases

*Received by the editor May 9, 2006; accepted for publication (in revised form) August 14, 2007; published electronically February 1, 2008.
http://www.siam.org/journals/sirev/50-1/65945.html

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

the bulge to the bottom of the matrix. Once the matrix has been returned to upper Hessenberg form, the iteration is complete:



The path from the basic QR algorithm to the bulge-chasing algorithms, which are also known as *implicitly shifted QR algorithms*, has quite a few steps and takes a while to traverse. It is probably fair to say that most classroom treatments of the QR algorithm never get that far. For example, [16] mentions implicitly shifted QR algorithms but does not discuss them.

It seems to me that it would be desirable to show students the algorithm that is actually used. Thus the following question arises: Can we carve a reasonable pedagogical path that leads straight to the bulge-chasing algorithms, bypassing (1.1) entirely? It is the object of this paper to sketch such a path.

2. Preparation. First of all, there is no magical route. No matter how you want to get to the QR algorithm, a fair amount of preparation is needed. In this section we outline the necessary preparatory steps, most of which would still be necessary even if we were planning to start at (1.1). The one bit that could have been avoided is the material on Krylov subspaces at the end of the section. One common approach is to introduce the explicit and implicit QR algorithms and then use the implicit-Q theorem [22, pp. 381–2] to show that they are equivalent. Krylov subspaces underlie the proof of the implicit-Q theorem, but they are usually not mentioned explicitly. I never liked this approach; it feels a bit like pulling a rabbit out of a hat. I prefer to get the Krylov subspaces out in the open. Not only does this show more clearly what is going on, it also allows an easy extension of the theory to more general classes of GR algorithms [23]. Moreover, it serves as preparation for what is often the next topic: Krylov subspace methods for solving large, sparse eigenvalue problems [10], [13], [14], [23].

Similarity Transformations. The student needs to know the basic theoretical facts about eigenvalues, including some fundamental ideas associated with similarity transformations. Matrices $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times n}$ are *similar* if there is a nonsingular $S \in \mathbb{C}^{n\times n}$ such that $B = S^{-1}AS$. Similar matrices have the same eigenvalues, and their eigenvectors are related in a simple way: If v is an eigenvector of A associated with eigenvalue λ, then $S^{-1}v$ is an eigenvector of B associated with λ. This is an easy consequence of the definition of eigenvector and the equation $B = S^{-1}AS$.

The student must also understand that a similarity transformation is a change of coordinate system. Similar matrices A and B represent the same linear transformation in two different coordinate systems. A vector whose coordinates are given by x in the "A" coordinate system has coordinates $S^{-1}x$ in the "B" coordinate system. This idea is reinforced by the observation that an eigenvector v of A corresponds to an eigenvector $S^{-1}v$ of B.

Invariant subspaces are important. A subspace $\mathcal{S}$ of $\mathbb{C}^n$ is *invariant* under A if, for every $x \in \mathcal{S}$, Ax is also in $\mathcal{S}$; briefly $A\mathcal{S} \subseteq \mathcal{S}$. If you can find an invariant subspace, then you can split your eigenvalue problem into smaller pieces, as the following proposition shows.

PROPOSITION 2.1. *Let $\mathcal{S}$ be a j-dimensional subspace ($1 \le j \le n-1$) that is invariant under A, let S be a nonsingular matrix whose first j columns are a basis for*

$\mathcal{S}$, and let $B = S^{-1}AS$. Then B is block upper triangular:

$$B = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$

where B_{11} is $j \times j$.

The eigenvalues of A are the same as those of B, which are the same as those of B_{11} and B_{22} together. Thus we have split the eigenvalue problem into the two smaller eigenvalue problems for B_{11} and B_{22}. Whenever you reduce a problem to smaller problems, you've made progress. If you can split those small problems into still smaller problems, and so on, you will eventually solve the eigenvalue problem completely. Therefore, if you can find invariant subspaces, you can solve eigenvalue problems. The easy proof of Proposition 2.1 can be found in [22, Theorem 6.1.6] and many other places.

Partition the matrix S from Proposition 2.1 as $S = \begin{bmatrix} S_1 & S_2 \end{bmatrix}$, where S_1 consists of the first j columns. A vector is in $\mathcal{S}$ if and only if it can be expressed as $S_1 z$ for some $z \in \mathbb{C}^j$. The equation $AS = SB$ implies that $AS_1 = S_1 B_{11}$. The next proposition is an immediate consequence.

PROPOSITION 2.2. *(λ, v) is an eigenpair of B_{11} if and only if $(\lambda, S_1 v)$ is an eigenpair of A.*

The eigenvalues of B_{11} are accordingly called the *eigenvalues of A associated with the invariant subspace $\mathcal{S}$.*

The Power Method and Extensions. So far we have been discussing strictly theoretical issues. Now we begin to think about computation. Every discussion of eigenvector computations begins with the power method. We will make the simplifying assumption that we are dealing with a matrix $A \in \mathbb{C}^{n\times n}$ that is *semisimple*, that is, it has a set of n linearly independent eigenvectors $v_1, v_2, \ldots, v_n$, associated with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, respectively. Assume these objects are numbered so that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|.$$

If $|\lambda_1| > |\lambda_2|$, then, for almost any choice of starting vector q, the sequence

$$q,\ Aq,\ A^2q,\ A^3q,\ \ldots, \tag{2.1}$$

if properly rescaled, will converge to a multiple of v_1, an eigenvector associated with the dominant eigenvalue λ_1. The convergence is linear with ratio $|\lambda_2/\lambda_1|$. This is the *power method.*

It is easy to see why the power method works. Since $v_1, \ldots, v_n$ are linearly independent, they form a basis of $\mathbb{C}^n$. Thus

$$q = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

for some (unknown) scalars $c_1, \ldots, c_n$. Except in extremely unlucky cases, $c_1 \neq 0$. Then

$$Aq = c_1\lambda_1 v_1 + c_2\lambda_2 v_2 + \cdots + c_n\lambda_n v_n,$$

and in general

$$\begin{aligned} A^j q &= c_1\lambda_1^j v_1 + c_2\lambda_2^j v_2 + \cdots + c_n\lambda_n^j v_n \\ &= \lambda_1^j \left(c_1 v_1 + c_2(\lambda_2/\lambda_1)^j v_2 + \cdots + c_n(\lambda_n/\lambda_1)^j v_n\right). \end{aligned}$$

If we ignore the irrelevant factor λ_1^j, we see that the component in the direction of v_1 stays fixed, while all other components tend to zero like $|\lambda_2/\lambda_1|^j$ or faster. Thus we have convergence to a multiple of v_1.

Each vector in (2.1) is just a representative of the one-dimensional subspace that it spans. When we rescale the vector, we are just replacing one representative (basis) of the subspace by another. Thus we can view (2.1) as a sequence of one-dimensional subspaces

$$\mathcal{S},\ A\mathcal{S},\ A^2\mathcal{S},\ A^3\mathcal{S},\ \ldots$$

that converges to the eigenspace $\text{span}\{v_1\}$.

This leads to an obvious generalization of the power method called *subspace iteration:* Given a subspace $\mathcal{S}$ of $\mathbb{C}^n$ of any dimension k, consider the sequence of subspaces

$$\mathcal{S},\ A\mathcal{S},\ A^2\mathcal{S},\ A^3\mathcal{S},\ \ldots. \tag{2.2}$$

If $|\lambda_k| > |\lambda_{k+1}|$, then for almost any choice of $\mathcal{S}$ with dimension k, all of the spaces in sequence (2.2) will have dimension k, and the sequence will converge to the invariant subspace $\text{span}\{v_1, \ldots, v_k\}$ corresponding to the dominant eigenvalues $\lambda_1, \ldots, \lambda_k$. Convergence is linear with ratio $|\lambda_{k+1}/\lambda_k|$.

This result remains true with minor modifications even if A is not semisimple. A careful proof requires some effort [23], [24]. One must first define a metric that measures the distance between subspaces, then establish some properties of the metric, and finally demonstrate convergence. However, a formal proof is hardly necessary on a first pass. It is easy to argue the plausibility of the result: Each $q \in \mathcal{S}$ can be expressed in the form

$$q = c_1v_1 + c_2v_2 + \cdots + c_nv_n.$$

Except for extremely unlucky choices of $\mathcal{S}$, no $q \in \mathcal{S}$ will have $c_1 = \cdots = c_k = 0$. We have

$$A^jq = \lambda_k^j \left[c_1(\lambda_1/\lambda_k)^jv_1 + \cdots + c_kv_k + c_{k+1}(\lambda_{k+1}/\lambda_k)^jv_{k+1} + \cdots + (\lambda_n/\lambda_k)^jv_n\right].$$

We see that the components of A^jq in $\text{span}\{v_1, \ldots, v_k\}$ are all either constant or growing, while the components outside of $\text{span}\{v_1, \ldots, v_k\}$ are all tending to zero at the rate of $|\lambda_{k+1}/\lambda_k|^j$ or faster. Thus the vectors in $A^j\mathcal{S}$ are all moving toward $\text{span}\{v_1, \ldots, v_k\}$ at the claimed rate. Since the limiting space must have dimension k, it must be all of $\text{span}\{v_1, \ldots, v_k\}$.

At this point it is advantageous to make a slight generalization. Let m be a small number such as 1, 2, 4, or 6 (for example), choose m *shifts* $\rho_1, \ldots, \rho_m \in \mathbb{C}$, and let

$$p(A) = (A - \rho_1I)(A - \rho_2I)\cdots(A - \rho_mI).$$

$p(A)$ has the same eigenvectors as A, and the corresponding eigenvalues $p(\lambda_1), \ldots, p(\lambda_n)$. Let us renumber these objects so that

$$|p(\lambda_1)| \geq |p(\lambda_2)| \geq \cdots \geq |p(\lambda_n)|.$$

Then if $|p(\lambda_k)| > |p(\lambda_{k+1})|$, the subspace iterations

$$\mathcal{S},\ p(A)\mathcal{S},\ p(A)^2\mathcal{S},\ p(A)^3\mathcal{S},\ \ldots \tag{2.3}$$

driven by $p(A)$ converge (for almost every choice of starting subspace $\mathcal{S}$) to the invariant subspace $\text{span}\{v_1, \ldots, v_k\}$ linearly with ratio $|p(\lambda_{k+1})/p(\lambda_k)|$. Depending upon the values of k and m, it may be possible to choose the shifts $\rho_1, \ldots, \rho_m$ in such a way that $|p(\lambda_{k+1})/p(\lambda_k)| \ll 1$, yielding rapid convergence. We will say more about this later.

The advantages of making this generalization are twofold. First, it brings shifts, which are essential to rapid convergence, into the picture. Second, it prepares the way for QR iterations of degree two or greater, which are the norm in practice.

If we want to carry out the subspace iterations (2.3) in practice, we need to operate on a basis for $\mathcal{S}$. Let

$$q_1^{(0)},\ q_2^{(0)},\ \ldots,\ q_k^{(0)} \tag{2.4}$$

be an orthonormal basis of $\mathcal{S}$. Then

$$p(A)q_1^{(0)},\ p(A)q_2^{(0)},\ \ldots,\ p(A)q_k^{(0)} \tag{2.5}$$

is a basis for $p(A)\mathcal{S}$, which we can then orthonormalize by some version of the Gram–Schmidt process to get an orthonormal basis

$$q_1^{(1)},\ q_2^{(1)},\ \ldots,\ q_k^{(1)} \tag{2.6}$$

for $p(A)\mathcal{S}$. Repeating this procedure we can obtain orthonormal bases

$$q_1^{(j)},\ q_2^{(j)},\ \ldots,\ q_k^{(j)} \tag{2.7}$$

for $p(A)^j\mathcal{S}$ for $j = 0, 1, 2, 3, \ldots$. This is called *simultaneous iteration*.

A highly advantageous characteristic of simultaneous iteration is that it automatically carries out subspace iterations not only on $\mathcal{S}$, but on subspaces of $\mathcal{S}$ as well. For each $i < k$, the first i vectors in (2.4) are an orthonormal basis of an i-dimensional subspace $\mathcal{S}_i = \text{span}\{q_1^{(0)}, \ldots, q_i^{(0)}\}$, and the first i vectors in (2.5) are a basis of $p(A)\mathcal{S}_i$. Moreover, since the Gram–Schmidt process preserves the subspaces spanned by leading vectors, the first i vectors in (2.6) also form a basis for $p(A)\mathcal{S}_i$. Since this relationship holds on every step, we see that for every j, the first i vectors in (2.7) form an orthonormal basis for $p(A)^j\mathcal{S}_i$. Thus simultaneous iteration automatically carries out subspace iterations on the nested spaces $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k = \mathcal{S}$ all at once.

Now think about doing simultaneous iteration on a set of n vectors

$$q_1^{(0)},\ q_2^{(0)},\ \ldots,\ q_n^{(0)}.$$

Then subspace iterations on the spaces $\mathcal{S}_k = \text{span}\{q_1^{(0)}, \ldots, q_k^{(0)}\}$ of dimension $k = 1, 2, \ldots, n$ are taking place simultaneously. If any one of these sequences converges rapidly to an invariant subspace $\text{span}\{v_1, \ldots, v_k\}$, then we will be in a position to split the problem into two smaller subproblems, as shown in Proposition 2.1. Therefore all of the ratios $|p(\lambda_{k+1})/p(\lambda_k)|$, $k = 1, \ldots, n-1$, are relevant. If any one of them is small, we will make rapid progress toward reducing our problem. Thus our objective is to choose shifts that make at least one of these ratios small.

Suppose we are able to find shifts that are good approximations to eigenvalues of A, say,

$$\rho_1 \approx \lambda_n,\ \rho_2 \approx \lambda_{n-1},\ \ldots,\ \rho_m \approx \lambda_{n-m+1}.$$

Think of the generic case, in which all of the eigenvalues are distinct, each ρ_i approximates one eigenvalue well, and no ρ_i is close to any of the other eigenvalues. Recalling that

$$p(z) = (z - \rho_1)(z - \rho_2) \cdots (z - \rho_m),$$

we see that $p(\rho_i) = 0$, $i = 1, \ldots, m$, so it must be that $|p(\lambda_{n-i+1})|$ is tiny if ρ_i is a good enough approximation to λ_{n-i+1}, $i = 1, \ldots, m$. Thus m of the values $|p(\lambda_k)|$ are tiny. The other $n - m$ values are not tiny, because no ρ_i is close to any of the other eigenvalues of A. Thus

$$|p(\lambda_{n-m})/p(\lambda_{n-m+1})| \ll 1,$$

and we have rapid convergence to the invariant subspace $\text{span}\{v_1, \ldots, v_{n-m}\}$.

The next obvious question is this: Where are we going to find good shifts? Here we provide a general answer, leaving a more specific answer for later. Initially we may have no idea of where the eigenvalues lie, but after a number of iterations some eigenvalues will begin to emerge. So far we have been speaking as if we are just going to pick some shifts and use the same shifts on each iteration. However, there is nothing to stop us from changing to better shifts as they become available. As soon as some good approximations of eigenvalues emerge, we can use them as shifts for subsequent iterations, thereby improving the convergence rate. With the more rapid convergence, we soon have much better approximations to eigenvalues, i.e., better shifts. Inserting these improved shifts, we improve the convergence rate still more, and so on.

The operations we have outlined here are very computationally intensive; the amount of arithmetic required on each step of simultaneous iteration on n vectors is $O(n^3)$, which we cannot afford. As we shall see, the implicit QR algorithm is a method that effects these very operations much more efficiently.

Upper Hessenberg Matrices. The key to efficiency is to work with Hessenberg matrices. A matrix H is *upper Hessenberg* if $h_{ij} = 0$ whenever $i > j + 1$. Because Hessenberg matrices are so close to triangular form, they are cheap to work with.

Every $A \in \mathbb{C}^{n\times n}$ is unitarily similar to a matrix in upper Hessenberg form. The similarity transformation can be effected by a sequence of $n-1$ reflectors (Householder transformations) [22] at a cost of $O(n^3)$ floating point operations (flops). The first reflector has the form

$$Q_1 = Q_1^* = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \tilde{Q}_1 & \\ 0 & & & \end{array}\right]$$

and creates the desired zeros in the first column. That is, demonstrating in the 5×5 case,

$$Q_1^* A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}.$$

Because of the form of Q_1, this first transformation leaves the first row of A unchanged. Then when Q_1 is applied on the right to complete the similarity transformation, the

first column is left untouched, so the zeros are left intact. The second reflector has the form

$$Q_2 = Q_2^* = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \tilde{Q}_2 & \\ 0 & 0 & & & \end{array}\right]$$

and creates the desired zeros in the second column. Q_3 creates zeros in the third column, and so on. After $n-2$ similarity transforms, we will have reached upper Hessenberg form. That is, if we let $Q = Q_1 Q_2 \cdots Q_{n-2}$, then the matrix

$$B = Q^* A Q$$

will be upper Hessenberg.

Notice that each of the reflectors satisfies $Q_j e_1 = e_1$, so $Qe_1 = e_1$ also. That is, the first column of Q is e_1. Actually we can modify the algorithm so that the first column of Q is proportional to any nonzero vector x of our choosing. Given x, let Q_0 be a reflector whose first column is proportional to x, i.e., $Q_0 e_1 = \beta x$, where $|\beta| = 1/\|x\|_2$. Begin the reduction by transforming A to $Q_0^* A Q_0$, then transform this matrix to upper Hessenberg form. Letting $Q = Q_0 Q_1 \cdots Q_{n-2}$, we have that $B = Q^* A Q$ is upper Hessenberg. Moreover $Qe_1 = Q_0 e_1 = \beta x$, so the first column of Q is proportional to x.

Because reflectors have the general form $U = I + uv^*$, they can be applied inexpensively. If U is $k \times k$ and W is $k \times j$, the transformation $W \to UW$ requires only about $4kj$ flops if it is organized as

$$UW = W + u(v^* W).$$

Similarly, if X is $j \times k$, the transformation $X \to XU$, organized as

$$XU = X + (Xu)v^*,$$

also takes about $4kj$ flops. Using these numbers and adding up the flops for the reduction to Hessenberg form, we find that the total flop count is about $\frac{16}{3}n^3$ flops, including the cost of assembling the matrix Q. The following proposition summarizes our findings.

PROPOSITION 2.3. *For $A \in \mathbb{C}^{n\times n}$ and nonzero $x \in \mathbb{C}^n$, there is a unitary $Q \in \mathbb{C}^{n\times n}$ such that $Qe_1 = \beta x$ for some $\beta \neq 0$ and $B = Q^* A Q$ is upper Hessenberg. Q and B can be computed directly in about $\frac{16}{3}n^3$ flops. Q is a product of $n-1$ reflectors.*

Krylov Subspaces. Our final preparatory step is to introduce Krylov subspaces, which are closely linked to Hessenberg matrices. Given $x \neq 0$, the jth *Krylov subspace* associated with A and x, denoted $\mathcal{K}_j(A, x)$, is defined by

$$\mathcal{K}_j(A, x) = \operatorname{span}\{x, Ax, A^2 x, \ldots, A^{j-1} x\}.$$

The proofs of the following lemmas are easy exercises. Let $e_1, \ldots, e_n$ denote the columns of the $n \times n$ identity matrix, as usual. An upper Hessenberg matrix is *properly upper Hessenberg* if $h_{ij} \neq 0$ whenever $i = j + 1$.

LEMMA 2.4. *If H is properly upper Hessenberg, then*

$$\mathcal{K}_j(H, e_1) = \text{span}\{e_1, \ldots, e_j\}, \quad j = 1, \ldots, n.$$

LEMMA 2.5. *If $x = p(A)e_1$, then*

$$p(A)\mathcal{K}_j(A, e_1) = \mathcal{K}_j(A, x), \quad j = 1, \ldots, n.$$

Last but not least, we establish the link between Krylov subspaces and properly upper Hessenberg matrices: in a similarity transformation to proper upper Hessenberg form, the leading columns of the transforming matrix span Krylov subspaces.

PROPOSITION 2.6. *Suppose $B = Q^{-1}AQ$ and B is properly upper Hessenberg. Let $q_1, \ldots, q_n$ denote the columns of Q. Then*

$$\text{span}\{q_1, \ldots, q_j\} = \mathcal{K}_j(A, q_1), \quad j = 1, \ \ldots, \ n.$$

Proof. The proof is by induction on j. The proposition holds trivially for $j = 1$. Rewrite the similarity transformation as $AQ = QB$. Equating jth columns of this equation, we have

$$Aq_j = \sum_{i=1}^{j+1} q_i b_{ij},$$

which we rewrite as

$$q_{j+1}b_{j+1,j} = Aq_j - \sum_{i=1}^{j} q_i b_{ij}. \tag{2.8}$$

Assuming inductively that $\text{span}\{q_1, \ldots, q_j\} = \mathcal{K}_j(A, q_1)$, we find that

$$Aq_j \in A\mathcal{K}_j(A, q_1) \subseteq \mathcal{K}_{j+1}(A, q_1).$$

Therefore, by (2.8), $q_{j+1} \in \mathcal{K}_{j+1}(A, q_1)$. Here it is crucial that $b_{j+1,j} \neq 0$. Since $q_1, \ \ldots, \ q_j, \ q_{j+1} \in \mathcal{K}_{j+1}(A, q_1)$, we have $\text{span}\{q_1, \ldots, q_{j+1}\} \subseteq \mathcal{K}_{j+1}(A, q_1)$. But $\text{span}\{q_1, \ldots, q_{j+1}\}$ has dimension $j + 1$ and $\mathcal{K}_{j+1}(A, q_1)$ has dimension at most $j + 1$. Therefore $\text{span}\{q_1, \ldots, q_{j+1}\} = \mathcal{K}_{j+1}(A, q_1)$. ∎

3. Description of the Implicitly Shifted *QR* Algorithm. Since any matrix can be reduced to upper Hessenberg form at $O(n^3)$ cost, we will assume at the outset that we are working with an $A \in \mathbb{C}^{n\times n}$ that is in upper Hessenberg form. Moreover, we will assume without loss of generality that it is in proper upper Hessenberg form. If it is not, then A is block upper triangular, and we can break the eigenvalue problem for A into two or more smaller subproblems for which the matrices are properly upper Hessenberg. We will describe a single implicit QR iteration of degree m, transforming the proper upper Hessenberg matrix A to an upper Hessenberg matrix $B = Q^*AQ$.

The iteration begins with the choice of m shifts $\rho_1, \ \ldots, \ \rho_m$. There are many strategies for choosing shifts, but a typical one is to compute the eigenvalues of the lower-right-hand $m \times m$ submatrix of A and use them as shifts. (Remember, m is small.)

Next a vector

$$x = \alpha(A - \rho_1 I)(A - \rho_2 I)\cdots(A - \rho_m I)e_1$$

is computed, where α is any convenient nonzero scale factor. This is an inexpensive computation because A is upper Hessenberg. First $x_1 = \alpha_1(A - \rho_m I)e_1$ is computed. This is proportional to the first column of $A - \rho_m I$, and all but its first two entries are zero. Next $x_2 = \alpha_2(A - \rho_{m-1}I)x_1$ is computed. It is a linear combination of the first two columns of $(A - \rho_{m-1}I)$, and all but its first three entries are zero. Continuing in this manner for m steps, we find that x is a linear combination of the first m columns of $A - \rho_1 I$, and all but its first $m+1$ entries are zero. The cost of computing x is just under $\frac{2}{3}m^3$ flops. Notice that

$$x = p(A)e_1,$$

where

$$p(A) = \alpha(A - \rho_1 I)(A - \rho_2 I)\cdots(A - \rho_m I).$$

Thus x is the first column of $p(A)$. We have obtained x cheaply without actually computing $p(A)$.

Define $\tilde{x} \in \mathbb{C}^{m+1}$ by $x = \left[\begin{smallmatrix}\tilde{x}\\ 0\end{smallmatrix}\right]$. Let $\tilde{Q}_0 \in \mathbb{C}^{(m+1)\times(m+1)}$ be a reflector such that $\tilde{Q}_0 e_1 = \beta\tilde{x}$ for some $\beta \neq 0$, and let

$$Q_0 = \begin{bmatrix} \tilde{Q}_0 & 0 \\ 0 & I_{n-m-1} \end{bmatrix}.$$

Then $Q_0 e_1 = \beta x$.

Begin a unitary similarity transformation by forming a new matrix $B_0 = Q_0^* A Q_0$. Because of the form of Q_0, the transformation $A \to Q_0^* A$ recombines only the first $m+1$ rows of A. This disturbs the upper Hessenberg form, but only in the first $m+1$ rows. There is now a bulge in the upper Hessenberg form. The transformation $Q_0^* A \to Q_0^* A Q_0 = B_0$ recombines the first $m+1$ columns. Because $a_{m+2,m+1} \neq 0$, one additional row gets added to the bulge. The transformed matrix has the form

$$B_0 = \left[\begin{array}{|cccc|}\hline \multicolumn{4}{|c|}{\multirow{3}{*}{$\ddots$}} & \\ & & & \\ & & & \\ \hline & & & \\ & & & \\ & & & \\ \hline \end{array}\right],$$

where all entries outside of the outlined region are zero. The tip of the bulge is at position $(m+2, 1)$.

The rest of the implicit QR iteration consists of returning the matrix to upper Hessenberg form by the method that we outlined in the previous section. This amounts to chasing the bulge. First a Q_1 is built such that the first column of $Q_1^* B_0$ is restored to upper Hessenberg form. Since only the entries $(3,1), \ldots, (m+2,1)$ need to be set to zero, Q_1 has the form $Q_1 = \operatorname{diag}\{1, \tilde{Q}_1, I_{n-m-2}\}$. Thus the transformation $B_0 \to Q_1^* B_0$ acts on rows $2, \ldots, m+2$ only. Then the transformation $Q_1^* B_0 \to B_1 = Q_1^* B_0 Q_1$ acts on columns $2, \ldots, m+2$. It leaves the newly created zeros intact, but it adds a new row to the bulge because $a_{m+3,m+2} \neq 0$. Therefore the bulge doesn't shrink, it gets moved. Each subsequent transformation removes one column from the bulge and adds a row, chasing the bulge down and to the right. In mid-chase we have

$$B_k = \left[\begin{array}{|cccc|}\hline \multicolumn{4}{|c|}{\multirow{3}{*}{$\ddots$}} & \\ & & & \\ & & & \\ \hline & & & \\ & & & \\ & & & \\ & & & \\ \hline \end{array}\right].$$

Eventually the bulge is chased off the bottom of the matrix, and Hessenberg form is restored. This completes the implicit QR step. The new iterate is $B = B_{n-2} = Q^*AQ$, where $Q = Q_0Q_1 \cdots Q_{n-2}$. Because $Q_1e_1 = e_1$, ..., $Q_{n-2}e_1 = e_1$, we have $Qe_1 = Q_0e_1 = \beta x$. Thus the first column of Q is proportional to x, which is the first column of $p(A)$.

The implicit QR iteration is an instance of the transformation in Proposition 2.3. It "reduces" A to upper Hessenberg form by a transformation whose first column is proportional to a specified vector x.

The implicit QR step is much less expensive then the general reduction to Hessenberg form, because each Q_i used in the bulge chase acts on only $m+1$ rows or columns. For example, the initial transformation $A \to Q_0^*A$ effectively acts on an $(m+1) \times n$ submatrix, so it costs about $4(m+1)n$ flops. The transformation $Q_0^*A \to Q_0^*AQ_0$ effectively acts on an $(m+2) \times (m+1)$ submatrix, so it costs about $4(m+1)(m+2)$ flops. Thus the cost of applying Q_0 is about $4(m+1)(n+m+2)$ flops altogether. One easily checks that each of the subsequent Q_i transformations also costs about $4(m+1)(n+m+2)$ flops, so the total flop count for the iteration is about $4(m+1)(n+m+2)(n-1)$. We have ignored the cost of building x, which at $O(m^3)$ is negligible, and the cost of building the reflectors, which at $O(m^2)$ per reflector is also negligible. Since n is large and m is small, we abbreviate the flop count to $4(m+1)n^2$. Since m is small and fixed, we can say that the cost of an iteration is $O(n^2)$.

Finally we remark that if A is real, then the entire algorithm stays within real arithmetic, provided that the shifts are chosen so that complex shifts come in conjugate pairs. This guarantees that $p(A)$ is real, x is real, the transformations Q_i can be chosen to be real, and the resulting B is real. In the real case we should always take $m \geq 2$ so that we can get complex shifts, which are needed to approximate complex eigenvalues.

4. What the Implicitly Shifted *QR* Algorithm Does. Now we come to the heart of the matter, and we come to it quickly because we are well prepared. B is upper Hessenberg, but it may fail to be properly upper Hessenberg. This is good news when it happens, because it means that we can split the problem into two or more subproblems. We could say more[1] about this case, but instead we will focus on the case when B is properly upper Hessenberg, which is what we always see in practice.

Assuming B is properly upper Hessenberg, Proposition 2.6 implies that the leading columns of Q span Krylov subspaces:

$$\text{span}\{q_1, \ldots, q_j\} = \mathcal{K}_j(A, q_1), \quad j = 1, \ldots, n,$$

where q_1, ..., q_n denote the columns of Q. Since q_1 is proportional to $x = p(A)e_1$, we deduce from Lemma 2.5 that $\mathcal{K}_j(A, q_1) = \mathcal{K}_j(A, x) = p(A)\mathcal{K}_j(A, e_1)$. Moreover, from Lemma 2.4, $\mathcal{K}_j(A, e_1) = \text{span}\{e_1, \ldots, e_j\}$. Therefore

$$p(A)\text{span}\{e_1, \ldots, e_j\} = \text{span}\{q_1, \ldots, q_j\}, \quad j = 1, \ldots, n.$$

This equation shows that the columns of Q are the result of one step of simultaneous iteration driven by $p(A)$, starting from the vectors e_1, ..., e_n.

[1] If the iteration could be done in exact arithmetic, B would have a zero on the subdiagonal if and only if at least one of the shifts is exactly an eigenvalue. Although we aspire to make the shifts as close to eigenvalues as possible, it almost never happens that a shift exactly equals an eigenvalue. Even if it did, B would fail to be reducible in practice due to roundoff errors. The "exact shift" case is discussed in detail in [23].

The similarity transformation $B = Q^*AQ$ is a change of coordinate system that transforms each vector x to Q^*x. Thus the vectors $q_1, \ldots, q_n$ are mapped back to the standard basis vectors $e_1, \ldots, e_n$, since $Q^*q_j = e_j$ for $j = 1, \ldots, n$.

In summary, the implicit QR step effects a nested subspace iteration driven by $p(A)$ on the spaces $\text{span}\{e_1, \ldots, e_j\}$, $j = 1, \ldots, n$. It also does a change of coordinate system that maps the resulting spaces back to $\text{span}\{e_1, \ldots, e_j\}$, $j = 1, \ldots, n$. This is the main message of the paper.

5. On Shift Selection and Convergence. In section 4 it was claimed the eigenvalues of the lower-right-hand $m \times m$ submatrix make good shifts. To see why this is so, imagine at first that we choose shifts $\rho_1, \ldots, \rho_m$ somehow and do QR iterations repeatedly using these same shifts to produce a sequence of matrices (A_k). The QR steps are effectively subspace iterations driven by a fixed matrix

$$p(A) = \alpha(A - \rho_1 I) \cdots (A - \rho_m I).$$

Because of the change of coordinate system at each step, this version of subspace iteration keeps the subspaces fixed and changes the matrix from one step to the next.

Suppose, as before, that the eigenvalues are numbered so that

$$|p(\lambda_1)| \geq |p(\lambda_2)| \geq \cdots \geq |p(\lambda_n)|.$$

For each j for which $|p(\lambda_j)| > |p(\lambda_{j+1})|$, the subspace $\text{span}\{e_1, \ldots, e_j\}$ gets closer and closer to being an invariant subspace of A_k as k increases. Since $\text{span}\{e_1, \ldots, e_j\}$ is invariant under A_k if and only if

$$A_k = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad A_{11} \in \mathbb{C}^{j \times j}, \tag{5.1}$$

we infer that convergence of the subspace iterations will imply convergence of (A_k) to the block triangular form (5.1). This happens not just for one choice of j but for all j for which $|p(\lambda_j)| > |p(\lambda_{j+1})|$.

Now consider the case $j = n - m$. If it happens that $|p(\lambda_{n-m})| > |p(\lambda_{n-m+1})|$, then $\text{span}\{e_1, \ldots, e_{n-m}\}$ will come closer and closer to the invariant subspace corresponding to eigenvalues $\lambda_1, \ldots, \lambda_{n-m}$. In the limit we will have the configuration (5.1), where the eigenvalues of A_{11} are $\lambda_1, \ldots, \lambda_{n-m}$, the eigenvalues associated with $\text{span}\{e_1, \ldots, e_{n-m}\}$ (which corresponds to the dominant invariant subspace $\text{span}\{v_1, \ldots, v_{n-m}\}$ in the original coordinate system). Thus the eigenvalues of A_{22} are $\lambda_{n-m+1}, \ldots, \lambda_n$.

Now suppose we have not quite converged yet but are getting close. Then we have

$$A_k = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where A_{21} has one nonzero entry, which is small. The eigenvalues of A_{22} are not $\lambda_{n-m+1}, \ldots, \lambda_n$, but they are close. At this point it makes sense to take the m eigenvalues of A_{22} as new shifts $\rho_1, \ldots, \rho_m$ for subsequent iterations. The new $p(z) = (z - \rho_1)(z - \rho_2) \cdots (z - \rho_m)$ will have $p(\lambda_{n-m+1}), \ldots, p(\lambda_n)$ all tiny, because each of these λ_j is approximated well by one of the ρ_i. On the other hand, none of

$p(\lambda_1), \ldots, p(\lambda_{n-m})$ will be small, because none of those λ_j is approximated well by any of the shifts. Thus the ratio $|p(\lambda_{n-m+1})/p(\lambda_{n-m})|$ will be much smaller than 1, and convergence will be accelerated. After a few more iterations, we will have much better approximations to $\lambda_{n-m+1}, \ldots, \lambda_n$, and we can use these as new shifts, accelerating the convergence even more.

We have just argued that once the iterations begin to converge, it is a good idea to use the eigenvalues of the lower-right-hand $m \times m$ submatrix as shifts. Two questions remain. How soon is it reasonable to begin choosing shifts in this manner? How often should new shifts be chosen? Experience has provided answers to these questions. (1) It is reasonable to use this shifting strategy right from the first iteration. (2) New shifts should be chosen for each iteration. At first the approximations to eigenvalues will be poor, and convergence will be slow. However, it usually does not take too many iterations before good shifts begin to emerge. Good shifts imply rapid convergence, which soon yields much better shifts, which give much faster convergence, and so on. The positive feedback loop thus established results in quadratic convergence.[2]

Once $a_{n-m+1,n-m}$ is small enough to be considered zero, the problem can be split apart. The small matrix A_{22} amounts to a batch of m eigenvalues that have been found. Subsequent iterations can be applied to the submatrix A_{11}, which will soon yield another batch of m eigenvalues[3] and reduction to a still smaller active submatrix.

Experience suggests that the number of iterations required to break off an eigenvalue or a small chunk of eigenvalues is $O(1)$, so the complete set of eigenvalues will be found in $O(n)$ iterations. Since each iteration costs $O(n^2)$ flops, the implicit QR algorithm is considered to be an $O(n^3)$ algorithm.

6. Recent Developments. Some recent developments are worth mentioning, even though they are tangential to the thrust of this article. Braman, Byers, and Mathias [2] have developed a version of the QR algorithm that uses large m in the sense that many shifts are chosen at once. On a large matrix, say, $n = 2000$, one hundred or so shifts will be chosen by computing the eigenvalues of the lower-right-hand 100×100 submatrix. This is done using a conventional QR routine with $m = 2$ as an auxiliary routine. But then the 100 shifts are not used to perform an implicit QR iteration of degree 100 (chasing a huge bulge), as that has been shown to be ineffective due to roundoff errors [20]. Instead the 100 shifts are used to perform 50 implicit QR iterations of degree 2. The 50 small bulges can be chased one after the other in tight formation. The similarity transformations can be accumulated and applied in bunches, allowing the use of level 3 BLAS [1]. This strategy actually increases the flop count but allows much more efficient cache use, resulting in substantially improved performance. Another new development, also due to Braman, Byers, and Mathias [3], is an aggressive early deflation strategy that allows the eigenvalues of a large matrix to be found in significantly fewer iterations. See [3] for details. Both of these innovations have been incorporated in the recently released LAPACK 3.1.

[2] This is a fair assessment of what almost always happens. However, examples of nonconvergence are known and have to be dealt with in practice. Dynamic shifting strategies such as the one discussed here lead to excellent performance in practice but also make global convergence analysis more difficult.

[3] This is an idealization. Eigenvalues don't always emerge in batches of m, because in practice not all of the m shifts will be equally good approximants of eigenvalues. If $m = 4$, for example, the eigenvalues will not always emerge in chunks of four. Frequently chunks of one, two, or three will break off.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999.

[2] K. Braman, R. Byers, and R. Mathias, *The multishift QR algorithm. Part* I: *Maintaining well-focused shifts and level* 3 *performance*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 929–947.

[3] K. Braman, R. Byers, and R. Mathias, *The multishift QR algorithm. Part* II: *Aggressive early deflation*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 948–973.

[4] J. J. M. Cuppen, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[5] I. S. Dhillon, *A New* $O(n^2)$ *Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. thesis, University of California, Berkeley, 1997.

[6] J. J. Dongarra and D. C. Sorensen, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. s139–s154.

[7] J. G. F. Francis, *The QR transformation: A unitary analogue to the LR transformation.* I., Comput. J., 4 (1961), pp. 265–271.

[8] J. G. F. Francis, *The QR transformation.* II, Comput. J., 4 (1961), pp. 332–345.

[9] V. N. Kublanovskaya, *On some algorithms for the solution of the complete eigenvalue problem*, U.S.S.R. Comput. Math. and Math. Phys., 3 (1961), pp. 637–657.

[10] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998. http://www.caam.rice.edu/software/ARPACK/index.html.

[11] B. N. Parlett, *The new qd algorithms*, Acta Numer., (1995), pp. 459–491.

[12] B. N. Parlett and I. S. Dhillon, *Relatively robust representations of symmetric tridiagonals*, Linear Algebra Appl., 309 (2000), pp. 121–151.

[13] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[14] D. C. Sorensen, *Numerical methods for large eigenvalue problems*, Acta Numer., (2002), pp. 519–584.

[15] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.

[16] D. S. Watkins, *Understanding the QR algorithm*, SIAM Rev., 24 (1982), pp. 427–440.

[17] D. S. Watkins, *Fundamentals of Matrix Computations*, 1st ed., John Wiley and Sons, New York, 1991.

[18] D. S. Watkins, *Some perspectives on the eigenvalue problem*, SIAM Rev., 35 (1993), pp. 430–471.

[19] D. S. Watkins, *QR-like algorithms—an overview of convergence theory and practice*, in The Mathematics of Numerical Analysis, J. Renegar, M. Shub, and S. Smale, eds., Lectures in Appl. Math. 32, AMS, Providence, RI, 1996, pp. 879–893.

[20] D. S. Watkins, *The transmission of shifts and shift blurring in the QR algorithm*, Linear Algebra Appl., 241/243 (1996), pp. 877–896.

[21] D. S. Watkins, *QR-like algorithms for eigenvalue problems*, J. Comput. Appl. Math, 123 (2000), pp. 67–83.

[22] D. S. Watkins, *Fundamentals of Matrix Computations*, 2nd ed., Wiley Interscience, New York, 2002.

[23] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.

[24] D. S. Watkins and L. Elsner, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.

[25] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford University, Oxford, 1965.