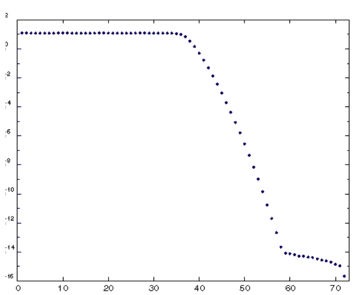
The QR algorithm

Striving for infallibility

by Cleve Moler

The QR algorithm is one of the most important, widely used, and successful tools we have in technical computation. Four variants of it are in MATLAB's mathematical core. They compute the eigenvalues of real symmetric matrices, eigenvalues of real nonsymmetric matrices, eigenvalues of pairs of complex matrices, and singular values of general matrices. These functions are used, in turn, to find zeros of polynomials, to solve special linear systems, to assess stability, and to perform many other tasks in various toolboxes.



Singular values of the Fourier matrix.

Dozens of people have contributed to the development of the various QR algorithms. But the first complete implementation and an important convergence analysis are due to J. H. Wilkinson. Wilkinson's book, *The Algebraic Eigenvalue Problem*, as well as two fundamental papers, were published in 1965, so it is reasonable to consider 1995 the thirtieth anniversary of the practical QR algorithm.

The QR algorithm is not infallible. It is an iterative process that is not always guaranteed to converge. So, on rare occasions, MATLAB users might see this message:

```
Error using ==> eig
Solution will not converge
```

A few years ago, recipients of this message might have just accepted it as unavoidable. But today, most people are surprised or annoyed; they have come to expect infallibility. So, we are still modifying our implementation, trying to improve convergence without sacrificing accuracy or applicability. We have recently made a couple of improvements that will be available in the next release of MATLAB, but which we can tell you about now.

The name "QR" is derived from the use of the letter Q to denote *orthogonal* matrices and the letter R to denote *right triangular* matrices. There is a `qr` function in MATLAB, but it computes the QR *factorization*, not the QR *algorithm*. Any matrix, real or complex, square or rectangular, can be factored into the product of a matrix Q with orthonormal columns and matrix R that is nonzero in only its upper, or right, triangle. You might remember the Gram Schmidt process, which does pretty much the same thing.

Using the `qr` function, a simple variant of the QR algorithm can be expressed as a MATLAB one-liner. Let `A` be a square, `n`-by-`n` matrix, and let `I = eye(n,n)`. Then one step of the QR iteration is given by

```
s = A(n,n);  [Q,R] = qr(A – s*I);  A = R*Q + s*I
```

If you enter this on one line, you can use the up-arrow key to iterate. The quantity `s` is the *shift*; it accelerates convergence. The QR factorization makes the matrix triangular.

$$A - sI = QR$$

Then the reverse order multiplication, RQ, restores the eigenvalues because

$$RQ + sI = Q'(A - sI)Q + sI = Q'AQ$$

so the new A is similar to the original A. Each iteration effectively transfers some "mass" from the lower to the upper triangle, while preserving the eigenvalues. As the iterations are repeated, the matrix often approaches an upper triangular matrix with the eigenvalues conveniently displayed on the diagonal.

For example, start with

```
A = gallery(3) =

  –149    –50   –154
   537    180    546
   –27     –9    –25
```

The first iteration,

```
 28.8263  –259.8671   773.9292
  1.0353    –8.6686    33.1759
 –0.5973     5.5786   –14.1578
```

is beginning to look upper triangular. After five more iterations we have

```
  3.0321    –8.0851   804.6651
  0.0017     0.9931   145.5046
 –0.0001     0.0005     1.9749
```

This matrix was contrived so its eigenvalues are equal to 1, 2, and 3. We can begin to see these three values on the diagonal. Eight more iterations give

```
   3.0716   -7.6952   802.1201
   0.0193    0.9284   158.9556
        0         0     2.0000
```

One of the eigenvalues has been computed to full accuracy and the below-diagonal element adjacent to it has become zero. It is time to *deflate* the problem and continue the iteration on the 2-by-2 upper left submatrix.

The QR algorithm is never practiced in this simple form. It is always preceded by a reduction to Hessenberg form, in which all the elements below the subdiagonal are zero. This reduced form is preserved by the iteration, and the factorizations can be done much more quickly. Furthermore, the shift strategy is more sophisticated, and is different for various forms of the algorithm.

The simplest variant involves real, symmetric matrices. The reduced form in this case is tridiagonal. Wilkinson provided a shift strategy that allowed him to guarantee convergence. Even in the presence of roundoff error, we do not know of any examples that cause the MATLAB implementation to fail.

The situation for real, nonsymmetric matrices is much more complicated. In this case, the given matrix has real elements, but its eigenvalues may well be complex. Real matrices are used throughout, with a double shift strategy that can handle two real eigenvalues, or a complex conjugate pair. Even thirty years ago, counterexamples to the basic iteration were known and Wilkinson introduced an “ad hoc” shift to handle them. But no one has been able to prove a complete convergence theorem.

We now know a 4-by-4 example that will cause the real, nonsymmetric QR algorithm to fail, even with Wilkinson’s ad hoc shift, but only on certain computers. The matrix is

```
A =
     0     2     0    -1
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

This is the companion matrix of the polynomial $p(x) = x^4 - 2x^2 + 1$, and the statement

```
roots([1 0 -2 0 1])
```

calls for the computation of `eig(A)`. The values $\lambda = 1$ and $\lambda = -1$ are both eigenvalues, or polynomial roots, with multiplicity two. For real x, the polynomial $p(x)$ is never negative. These double roots slow down the iteration so much that, on some computers, the vagaries of roundoff error interfere before convergence is detected. The iteration can wander forever, trying to converge but veering off when it gets close.

Similar behavior is shown by examples of the form

$$\begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & -\delta & 0 \\ 0 & \delta & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix}$$

where δ is small, but not small enough to be neglected, say $\delta = 10^{-8}$. The exact eigenvalues

$$\pm (1 - \delta^2/4)^{1/2} \pm i\,\delta/2$$

are close to a pair of double roots. The Wilkinson double shift iteration uses one eigenvalue from each pair and does change the matrix, but not enough to get rapid convergence.

The situation is not quite the same as the one which motivated Wilkinson’s ad hoc shift. If the computations were done exactly, or if we weren’t quite so careful about what we regard as negligible, we would eventually get convergence. But we are much better off if we modify the algorithm in a way suggested by Prof. Jim Demmel of U. C. Berkeley. The idea is to introduce a new, ad hoc shift if Wilkinson’s various shifts fail to get convergence. For this example, if we take a double shift based on repeating one of the eigenvalues of the lower 2-by-2 blocks, we get immediate satisfaction.

Our second case of QR failure, supplied by Prof. Alan Edelman of MIT, involves the SVD algorithm and a portion of the Fourier matrix. For example, take the upper left quarter of the Fourier matrix of order 144.

```
n = 144
F = fft(eye(n,n));
F = F(1:n/2,1:n/2)
s = svd(F)
semilogy(s,'.')
```

All the elements of F are complex numbers of modulus one. You can see from the graph that about half the singular values of F are equal to $\sqrt{n}$. (Edelman can explain why that happens.) The other half should decay rapidly toward zero, except that roundoff error intervenes. The computed values of the last dozen or so singular values do not show the continued decay that we expect. This is OK; these tiny singular values were destined to inaccuracy by the finite precision of the original matrix elements. The difficulty is that we worked harder than we should have to compute those small values. For some other values of `n`, the `svd` calculation fails to converge.

The SVD variant of the QR algorithm is preceded by a reduction to a bidiagonal form which preserves the singular values. For these Fourier matrices, the trailing portion of the two nonzero diagonals consists entirely of elements that are on the order of roundoff error in the original matrix elements, and that are not accurately determined by the original elements. Nevertheless, the current implementation of the bidiagonal QR iteration tries to accurately diagonalize the entire matrix. In effect, it tries to carry out an unshifted iteration on a portion of the matrix that has lost all significant figures. Convergence is chaotic and, in some cases, the iteration limit is reached. We plan to add a test that will avoid this behavior, but that will not jeopardize accuracy for matrices where the bidiagonal form retains accurate elements. ■

Cleve Moler is chairman and co-founder of The MathWorks. His e-mail address is moler@mathworks.com.