The QR Algorithm Beresford Pratt 2/00 PhL 12.17.04

This is a wonderful little 5-page paper I found on the web and printed and read. It has a nice mix of history and method.

History. First, people got more interested in matrices with QM in the 1920-30's. At that time, the method of doing things was (1) to compute the coeffs of the char poly and (2) solve it for the zeros. That gave you the eigenvalues. This two-stage method fails for n > 10 because the resulting roots are **too sensitive** to computation errors in the coefficients. Crushing N^2 numbers down to N was too much.

On page 39 we have some comments about matrix **stability** and how this requires that you maintain norms of things small. As you do similarities to get into upper Heisenberg form, you never want the norm of the overall matrix to get much bigger than what you started with, else error! I think this is what the balance routine does that you do before going to upper Hessenberg.

The **Lapack** routine is DHSEQR and it is the latest and greatest QR as of Feb 2000.

The LU and QR Algorithms.

You can iterate either one in fact in the same manner. But QR always exists, so we use it instead of LU. The Theorem 1 makes the big claim that the QR algorithm converges to an UT form known as the **Schur Form** where the diagonal eigenvalues are sorted from large to small down the diagonal. The algorithm "pushes" them into that order! A reference for proof of this big theorem are given. I find the results rather amazing.

He then shows that Hessenberg form is invariant under each iteration, as I now know. This is called the basic QR algorithm. One trick used in it is that you examine the (n,n-1) element for smallness. When it gets small enough, you pop out (n,n) as an eigenvalue (I guess it must be the smallest one), and you **deflate** the matrix by killing off the last row and column, and then you keep going.

He simply comments that **shifts** make it go faster, but he does not say how to pick the shifts.

In general, you need about 10n QR iterations or less and you are done for eigenvalues.

He then does a nice job showing how Francis in 1961 did his trick to handle **complex pairs** of eigenvalues with the "first column of Q" gizmo. You basically do a **double-iteration step** as shown That is, you go directly from iteration B_1 to B_3 and bypass the middle step where things would have complex numbers. He mentions that this involves pushing a certain **"bulge"** down and off the end of the candidate B_3 matrix until if falls off the end and you really do get Hessenberg form back again. Code cost is $5n^2$ operations.

Sometimes in general you may have to inject random shifts to avoid **cycling** loops.

In the **symmetric** case, Hessenberg means **tridiagonal** and instead of being $O(n^2)$ the QR is **O(n)**. This is where we get the **classical estimate of $10n^2$** as the cost of grinding a tridiagonal down to eigenvalues. He says the people believe convergence is cubic, but can only prove it is quadratic! This is the same as the cost of doing "a few matrix multiplies".

History Again.

Recall in Scheid reading about the **Quotient-Difference** method of finding roots of a polynomial. You made little columns of d's and e's and it was an iterative method that converged onto the roots, using a table with rhombus rules. The guy who did that was **Rutishauser** (Swiss 1954) whose PhD was the Q-D

thing. He saw that each step of his iteration was in fact that same kind of reversal of factors that we now see in the LU and QR methods, so he was the first to see why you might want to iterate in this strange way. He himself noted the LU Algorithm. Since LU does not always work (as we know), **Francis** and his mentor **Strachey** in 1961 found the QR algorithm, but Russian **Vera Kublan**... was also there in 1961. Francis gets credit for the double-step idea.

Final comments are that in 1955 it was a pain to find eigenvalues of a non-symmetric matrix, but by 1965 the method was routine thanks to Francis. People think of QR for matrix orders of n = few hundred, but there is now interest in working with monstrous "sparse matrices" with n = 300,000 say, and QR does not work at all in this realm. Among other things, it destroys sparsity.

So the QR Algorithm makes the "top ten" list that this paper is part of. These are listed at: http://www.csit.fsu.edu/~burkardt/fun/misc/algorithms_dongarra.html. I presume there is a nice little article about each one on the web. Mine is in Computing in Science and Engineering.

Jack Dongarra and Francis Sullivan published a list of "The Top Ten Algorithms of the Century." Their list included:

1. the **Monte Carlo** method or Metropolis algorithm, devised by John von Neumann, Stanislaw Ulam, and Nicholas Metropolis;
2. the **simplex method** of linear programming, developed by George **Dantzig**;
3. the **Krylov Subspace Iteration method**, developed by Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos;
4. the **Householder matrix decomposition**, developed by Alston Householder;
5. the **Fortran compiler**, developed by a team lead by John Backus;
6. the **QR algorithm** for eigenvalue calculation, developed by J Francis;
7. the **Quicksort algorithm**, developed by Anthony Hoare;
8. the **Fast Fourier Transform**, developed by James Cooley and John Tukey;
9. the **Integer Relation Detection Algorithm**, developed by Helaman Ferguson and Rodney Forcade; (given N real values X_I, is there a nontrivial set of integer coefficients A_I so that sum (1 <= I <= N) $A_I * X_I = 0$?
10. the **fast Multipole algorithm**, developed by Leslie Greengard and Vladimir Rokhlin; (to calculate gravitational forces in an N-body problem normally requires N^2 calculations. The fast multipole method uses order N calculations, by approximating the effects of groups of distant particles using multipole expansions)