Homework 12, Math 6610-1, Due Nov. 28

1. Why don't we consider the Rayleigh quotient for arbitrary matrices? The following will explain. Here we assume $A \in R^{n\times n}$.

   (a) For the Rayleigh quotient $\rho = \frac{x^T Ax}{x^T x}$, show that there exists another matrix that will result in the same Rayleigh quotient for any $x \neq 0$.

   (b) Every A can be decomposed into a symmetric and an antisymmetric parts:

   $$A = \frac{1}{2}(A + A^T) + \frac{1}{2}(A - A^T).$$

   Show that ρ depends on the symmetric part only which means part of the information in A is lost when you use the Rayleigh quotient for nonsymmetric A.

   (c) The success of the Rayleigh quotient iteration for symmetric matrices relies on the fact that its accuracy for eigenvalues is quadratic. When A is nonsymmetric, it is no longer true and we only have

   $$\rho(x) - \rho(q) = O(||x - q||)$$

   as $x \to q$ for an eigenvector q. Show this by verifying that $\nabla\rho(q) \neq 0$.

2. In this project, we are going to build a simple MATLAB code to find all the eigenvalues of a real symmetric matrix.

   (a) Write a function `T = tridiag(A)` that reduces a real symmetric $n \times n$ matrix to tridiagonal form by orthogonal similarity transformations. The information about output matrix `T` should be contained in a $m \times 2$ array, that is, only one of the sub and super diagonals is stored.

   (b) Write a function `Tnew = qralg(T)` that runs the unshifted QR algorithm on a real tridiagonal matrix `T`. You should use Givens rotations to find QR decompositions for `T`. Your program should stop and return the current tridiagonal matrix `T` as `Tnew` when the $n, n-1$ entry $|t_{n,n-1}| < 10^{-12}$.

   (c) Write a driver program which (i) calls `tridiag`, (ii) calls `qralg` to get one eigenvalue, (iii) calls `qralg` with a smaller matrix to get another eigenvalue, and so on until all of the eigenvalues of A are determined. Set things up so that the values of $|t_{n,n-1}|$ at every QR iteration are stored in a vector and so that at the end, your program generates a semilogy plot of these values as a function of the number of QR factorizations. This n will move from the original `length(A)` down to 2, where you can solve the eigenvalue problem by the quadratic formula. Try this on `A=hilb(4)`. The output should be a set of eigenvalues and a "sawtooth plot".

(d) Modify `qralg` so that it uses the Wilkinson shift at each step. Turn in the new sawtooth plot for the same example.

(e) Rerun your program for the matrix `A=diag(15:-1:1) + ones(15,15)` and generate two sawtooth plots corresponding to shift and no shift. Discuss the rates of convergence observed here for the earlier matrix. Is the convergence linear, superlinear, quadratic, cubic...? Is it meaningful to speak of a certain "number of QR iterations per eigenvalue?"