CS 6210 Assignment 5   Due: 11/13/15 (Fri) at 6pm

Scoring for each problem is on a 0-to-5 scale (5 = complete success, 4 = overlooked a small detail, 3 = good job on half the problem, 2 = OK job on half the problem, 1 = germ of a relevant solution idea, 0 = missed the point of the problem.) Independent of this, one point will be deducted for insufficiently commented code. Test code and related material are posted on the course website `http://www.cs.cornell.edu/courses/cs6210/2015fa/`. All solution M-Files must be submitted through the CMS system. You are allowed to discuss *background* issues with other students, but the codes you submit must be your own.

Topics: Eigenvalue Problems for symmetric, skew-symmetric and orthogonal matrices, Tridiagonal Methods, Jacobi Methods, the Lanczos Method.

1 An Orthogonal Matrix Eigenvalue Problem

Complete the following function so that it performs as specified:

```
  function k = nEigs(U,alfa,beta)
% U is an nxn real orthogonal matrix.
% If -1 < alfa < beta < 1, then k estimates the number of U's eigenvalues that
%          have real parts in the interval [alfa,beta].
% If alfa = beta = 1, then k estimates the number of U's eigenvalues that
%     are equal to one.
% If alfa = beta = -1, then k estimates the number of U's eigenvalues that
%     are equal to minus one.
% All other alfa-beta combinations are illegal, e.g., alfa = 0, beta = 1
```

Note that the eigenvalues of U are on the unit circle. It follows that if $Uz = \lambda z$ then $(\lambda + 1/\lambda)/2 = \mathrm{Re}(\lambda)$ is an eigenvalue of $A = (U + U^{-1})/2 = (U + U^T)/2$, a symmetric matrix. Make effective use of `hess` and the Sturm sequence function `nLess` that is provided. To force issues, your implementation is NOT allowed to use `schur` or `eig` or any other MATLAB eigensolver. Throughout your implementation you are "allowed" to set a matrix element to zero if its absolute value is 10^{-12} or less. An interesting way to generate test examples is `[U,R] = qr(randn(n,p))` with `p<n`. Submit `nEigs` to CMS.

2 3-by-3 Eigenvalue decomposition for Skew-Symmetric Matrices

If $A \in \mathbb{R}^{n\times n}$ is skew-symmetric, then $A^T = -A$ and all its eigenvalues are on the imaginary axis. The *real Schur decomposition* states that there is a real orthogonal matrix Q such that

$$Q^T AQ \;=\; \mathrm{diag}(D_1, \ldots, D_p)$$

where each D_k is either the 1-by-1 matrix 0 or a 2-by-2 matrix of the form

$$D_k = \left[\begin{array}{cc} 0 & \mu_k \\ -\mu_k & 0 \end{array}\right]$$

Note that in the latter situation $\lambda(D_k) = \{+i\cdot\mu_k, -i\cdot\mu_k\}$. Also observe that if n is odd then there must be a nonzero real vector z so $Az = 0$. (Why?) Complete the following function so that it performs as specified:

```
  function Q = RealSchur3(A)
% A is a 3x3 real skew-symmetric matrix.
% Q is a 3x3 orthogonal matrix so that Q'*A*Q has the form
%                  Q'*A*Q = [ 0  mu  0 ; -mu 0 0 ; 0 0 0]
```

Hint. Find a null vector z for A and a Householder matrix P that can zero all but one entry of z. What can you say about the structure of P^TAP? You are not allowed to use `eig` or `schur`. You are free to use `House` (see A4). Submit `RealSchur3` to CMS.

3 A Jacobi Procedure for Skew Symmetric Matrices

Read about the cyclic Jacobi idea in §8.5.4 and about the block Jacobi idea in §8.5.6. In this problem you are to implement a cyclic block Jacobi procedure for skew-symmetric matrices:

```
     function [Q,D,BlockOffTrace] = SkewJacobi(A,tol,MaxNumSweeps)
% A is an nxn skew symmetric matrix and n = 2m+1.
% Q is nxn orthogonal.
% D is nxn and block diagonal with 1x1 and 2x2 diagonal blocks.
% BlockOff(D) <= tol*norm(A,'fro') where D = Q'*A*Q.
% BlockOffTrace is a column vector with the property that BlockOffTrace(k) is
% the value of BlockOff(A) after k-1 sweeps.
```

The procedure should be based on this blocking:

$$A \;=\; \left[\begin{array}{ccccc} A_{11} & A_{12} & \cdots & A_{1m} & A_{1,m+1} \\ A_{21} & A_{22} & \cdots & A_{2m}, & A_{2,m+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mm} & A_{m,m+1} \\ A_{m+1,1} & A_{m+1,2} & \cdots & A_{m+1,m} & A_{m+1,m+1} \end{array}\right].$$

The blocks $A_{1,m+1}, \ldots, A_{m,m+1}$ are 2-by-1. The blocks $A_{m+1,1}, \ldots, A_{m+1,m}$ are 1-by-2. The block $A_{m+1,m+1}$ is 1-by-1. All other blocks are 2-by-2.

If a 2-by-2 (block) subproblem involves a block from the last block column or block row, then a 3-by-3 real Schur decomposition needs to be computed and you should use `RealSchur3`. If not, then a 4-by-4 real Schur decomposition needs to be found and you should use the MATLAB `schur` function.

After a sweep you should evaluate the comparison

$$\texttt{BlockOff}(A) \;\equiv\; \sqrt{\sum_{i\neq j} \| A_{ij} \|_F^2} \;\leq\; \text{tol} \cdot \| A \|_F$$

The iteration should terminate if this is true or if the number of completed sweeps equals `MaxNumSweeps`. Submit `SkewJacobi` to CMS.

4 Eigenvalues of Diagonal + Rank-1 Via Lanczos

The eigenvalues (and eigenvectors) of a symmetric matrix that is diagonal-plus-rank-one is an $O(n^2)$ computation. See GVL4 §8.4.3. The method outlined there involves a highly-structured rational function whose roots are the desired eigenvalues. In this problem you are to implement a Lanczos-based procedure for computing specified subsets of the eigenvalues:

```
   function eValues = SpecialEig(d,v,k,what)
  % d is a column n-vector with distinct nonzero entries
  % v is a column n-vector with nonzero entries
  % k is a positive integer that satifies 1<=k<n
  % Let A = diag(d) + vv'.
  % If what equals 'SA', then eValues is a column k-vector comprised
  %          of the k algebraically smallest eigenvalues of A.
  % If what equals 'LA', then eValues is a column k-vector comprised
  %          of the k algebraically largest eigenvalues of A.
  % If what equals 'SM', then eValues is a column k-vector comprised
  %          of the k smallest eigenvalues of A in magnitude
  % If what equals 'LM', then eValues is a column k-vector comprised
  %          of the k largest eigenvalues of A in magnitude.
  % In each case, eValues(1) < eValues(2) < ... < eValues(k).
```

You are to make effective use `eigs` for all eigenvalue computations. Start with `help eigs` and figure out the proper calling sequence. Submit `SpecialEig` to CMS. A test script `P4Grade` is available on the course website.