CS 621: Assignment 6

Due: Friday, December 2, 2005 (In Lecture or 4130 Upson by 4pm)

Scoring for each problem is on a 0-to-3 scale (3 = complete success, 2 = overlooked a small detail, 1 = germ of the right idea, 0 = missed the point of the problem.) One point will be deducted for insufficiently commented code. Unless otherwise stated, you are expected to utilize fully MATLAB's vectorizing capability subject to the constraint of being flop-efficient. Test drivers and related material are posted on the course website `http://www.cs.cornell.edu/courses/cs621/2005fa/`. For each problem submit output and a listing of all scripts/functions that you had to write in order to produce the output. You are allowed to discuss *background* issues with other students, but the codes you submit must be your own.

P1. (4-by-4 Schur Decomposition)

If $A \in \mathbb{R}^{4\times 4}$ has two pairs of complex conjugate eigenvalues, then a *normalized* real Schur decomposition has the form

$$U^T A U = \begin{bmatrix} \lambda_1 & \alpha_1 & c_{11} & c_{12} \\ \beta_1 & \lambda_1 & c_{21} & c_{22} \\ 0 & 0 & \lambda_2 & \alpha_2 \\ 0 & 0 & \beta_2 & \lambda_2 \end{bmatrix} \qquad \alpha_1\beta_1 < 0,\ \alpha_2\beta_2 < 0$$

where U is orthogonal. Note that $\lambda(A) = \{\, \lambda_1 \pm i\sqrt{-\alpha_1\beta_1}, \lambda_2 \pm i\sqrt{-\alpha_2\beta_2}\,\}$. Complete the following function so that it performs as specified.

```
  function [U,sep] = Schur4(T)
% T is a normalized Schur form of a 4-by-4 matrix A that has a pair of complex
%    conjugate eigenvalues.
% U is an orthogonal matrix such that U'*T*U = S is a normalized real schur
%    decomposition with S(1:2,1:2) having the same eigenvalues as T(3:4,3:4) and
%    S(3:4,3:4) having the same eigenvalues as T(1:2,1:2).
% sep is the smallest singular value of the linear transformation phi where
%    phi(X) = T(1:2,1:2)X - XT(3:4,3:4)
```

You are not allowed to use any MATLAB eigenvalue decomposition function such as `eig`, `schur` or `ordschur`. However, you may use `svd` to compute `sep`. (It's the smallest singular value of a certain 4-by-4 matrix.)

Submit a brief write-up that derives your method and explains its behavior in floating point arithmetic. (E.g., $\| \tilde{U}^T * \tilde{U} - I \| \approx$ `eps` where $\tilde{U}$ is the computed U.) It should also include a listing of your implementation. Submit you code electronically.

P2.(Block Hessenberg Reduction)

Suppose $A \in \mathbb{R}^{n\times n}$ and (for simplicity) $n = mr$. We say that $U^T A U = H$ is a block Hessenberg reduction if U is orthogonal and H is upper Hessenberg when regarded as an m-by-m matrix with r-by-r blocks, e.g.,

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ 0 & H_{32} & H_{33} & H_{34} \\ 0 & 0 & H_{43} & H_{44} \end{bmatrix}$$

with $H_{ij} \in \mathbb{R}^{r\times r}$. Complete the following function so that it performs as specified

```
  function [U,H] = BlockHess(A,U1)
% A is n-by-n, U1 is n-by-r with orthonornal columns, n is a multiple of the integer r.
% U is orthogonal, U(:,1:r) = U1, and U'*A*U = H is block upper Hessenberg
% with r-by-r blocks.
```

A "production" implementation would use Householder transformations. However, to make things easy, anytime in your implementation that you need an orthogonal $P \in \mathbb{R}^{\mu \times \mu}$ that upper triangularizes a given $X \in \mathbb{R}^{\mu \times r}$, just use `[P,R] = qr(X)`. And feel free to use P in its explicit form, e.g., `B = P'*B*P`. This adds an order of magnitude of work compared to working with P as a product of Householder transformations, but the boss has mellowed out! Submit a listing of your implementation and the output obtained by testing it with the script `P2`.

GTD6. (Block Diagonalization)

If

$$T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \qquad \text{and} \qquad Y_0 = \begin{bmatrix} I & X_0 \\ 0 & I \end{bmatrix}$$

then

$$Y_0^{-1} T Y_0 = \begin{bmatrix} T_{11} & T_{11}X_0 - X_0 T_{22} + T_{12} \\ 0 & T_{22} \end{bmatrix}$$

If T_{11} and T_{22} have no eigenvalues in common, then it is possible to zero the (1,2) block by solving the Sylvester equation $T_{11}X_0 - X_0T_{22} = -T_{12}$. The process can be repeated on T_{11} and T_{22} turning *them* into block diagonal form, e.g.

$$Y_1^{-1} T_{11} Y_1 = \begin{bmatrix} S_{11} & 0 \\ 0 & S_{22} \end{bmatrix} \qquad Y_1 = \begin{bmatrix} I & X_1 \\ 0 & I \end{bmatrix}$$

$$Y_2^{-1} T_{22} Y_2 = \begin{bmatrix} S_{33} & 0 \\ 0 & S_{44} \end{bmatrix} \qquad Y_2 = \begin{bmatrix} I & X_2 \\ 0 & I \end{bmatrix}$$

It follows that if

$$Y = \begin{bmatrix} I & X_0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & X_1 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & X_2 \\ 0 & 0 & 0 & I \end{bmatrix}$$

then $Y^{-1}TY = \text{diag}(S_{11}, S_{22}, S_{33}, S_{44})$. If T has distinct eigenvalues then we can ultimately diagonalize it by repeating the "(1,2) block zeroing" idea.

The problem is, we may have to resort to very ill–conditioned Y matrices and that is bad. Interesting question: How much can we block diagonalize subject to the constraint that $\kappa(Y_{final}) <= M$ for some constant M, e.g., $M = 10^6$. Precise version: Maximize the number of diagonal blocks subject to $\kappa_\infty(Y_{final}) <= M$. Complete the following function so that it performs as specified:

```
  function [Y,D] = BlockDiag(T,M)
% T is an n-by-n upper triangular matrix with T(1,1)>T(2,2)>...>T(n,n).
% Y is an n-by-n matrix so that D = Y^{-1}TY is block diagonal.
% Y is chosen to approximately maximize the number of diagonal blocks in D
% subject to the constraint that cond(Y,inf) <= M.
```

Implementation comments:

- This may be a setting where recursion makes sense.
- A condition estimator and a Sylvester equation solver will be provided on the website.
- You can zero out a block in the "current" T's upper triangle part, tentatively update the current Y, and accept or reject the zeroing based on the condition of the (tentatively) updated Y.
- In general, the condition of your current Y gets worse as more blocks are zeroed. Don't press the upper bound M too much during the early stages.
- Your algorithm should be $O(n^3)$.
- Y_{final} is a product of Y_{little}'s that have the form `[I X;0 I]`. Note that the condition of a Y_{little} is $(1+\| X \|_\infty)^2$. Updating the "current" Y by a Y_{little} is highly structured–can this structure be exploited?
- $\kappa(AB) \leq \kappa(A)\kappa(B)$ may be useful in your condition monitoring,

Submit your implementation electronically and a listing and derivation with the rest of the assignment.