CSci 5304 HW3 Due: Oct 27 2014

- This assignment is a mix of handwritten questions and Matlab coding questions. You should hand in everything on paper, including a printout of your matlab functions and scripts.
- You are encouraged to do this in pairs: hand in one copy of the entire assignment with both names on it. You should still satisfy yourself that you could answer every question on your own, since similar questions will appear on future exams.

1. Consider the matrices

$$A = \begin{pmatrix} 1 & -1 \\ 1 & -1.00001 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & -1 \\ -1 & 1.00001 \end{pmatrix}$$

What is ratio of the largest to smallest eigenvalues (in modulus) for A and for B? Show that $\kappa_2(A) = \kappa_2(B)$. What can you conclude about the ratio of the largest to smallest eigenvalues as a way of estimating sensitivity of a linear system? Would you consider A ot be well-conditioned or ill-conditioned?

2. (a) Find the LU factorization of the matrix:

$$A = \begin{pmatrix} 2 & 0 & 5 & 8 \\ 0 & 2 & -1 & -3 \\ -2 & 6 & 2 & -3 \\ 4 & -4 & 0 & 2 \end{pmatrix}$$

(b) Find the PA=LU factorization of A using partial pivoting.

(c) What is the determinant of A?

(e) Using the LU factors obtained in (a) find the second column of the inverse of A, without computing the whole inverse.

3. (a) Show that if A is Symmetric Positive Definite (SPD) then Trace$(AX) > 0$ for all SPD matrices X. (b) Show that if Trace$(AX) \geq 0$ for all Symmetric Positive Semi-Definite (PSD) matrices X then A is PSD.

4. Write Matlab functions to carry out Gaussian Elimination without pivoting to solve a linear system $Ax = b$ where A is a tridiagonal matrix, stored as three columns (the subdiagonal, the main diagonal, and the superdiagonal), and b is a vector of all ones of appropriate dimension. You'll need to write at least two functions, one to carry out Gaussian Elimination on A, together with b, and one to solve the resulting triangular system.

Your matlab function should avoid storing the matrix as a full matrix or sparse matrix, though you can use one of these to check your answers with the Matlab builtin functions. Apply this function to the matrix formed by the following Matlab expression:

```
m = 10 or 500 or something bigger;
A_trid = [ [nan; ones(2*m,1)*m ], (m+1+(-m:m)') , [ones(2*m,1)*m;nan]];
% the following converts the tridiagonal to a full matrix
% (just for the purpose of checking your answers)
A_full = diag(A_trid(:,2))+diag(A_trid(2:end,1),-1)+diag(A_trid(1:end-1,3),1);
```

The right hand side for this will be a vector of all ones of dimension $n = 2m + 1$, e_n.

Do all this for $m = 500 : 500 : 5000$ and time the elapsed time or CPU time for solving the linear system. To time the process, you can use the Matlab functions `tic,toc`, `etime`, or `cputime`. Submit a table of CPU or elapsed times together with norms of the residuals. What seems to be the complexity of solving tridiagonal linear systems? Is it $O(n)$, $O(n^2)$ or $O(n^3)$? Optionally, solve the linear systems with Matlab explicitly and compare the accuracy and costs of the two different solutions. Are they close or not?

5. Write a Matlab code that computes the QR factorization of a tridiagonal matrix A stored in the same compact form as given in the previous question, to solve the same system $Ax = b$ with b as before as well. The result should be $QR = A$, with R stored in the similar compact form as A (need additional space for one extra superdiagonal), and Q should be left as a sequence of Givens rotations or 2×2 Householder transformations. You can apply the Givens rotations or reflections to b as you go, to obtain $Q^T b$, and then solve the triangular system $Rx = Q^T b$ with a triangular system solver similar to that used in the previous question. In this case, the upper triangular will have two non-zero diagonals above the main diagonal, instead of just one.

   Each Givens rotation of the form

$$G_{ij}(c,s) = \begin{array}{cccccccc|l}
 & & & \overset{\text{col } i}{\downarrow} & \overset{\text{col } j}{\downarrow} & & & & \\
1 & & & & & & & & \\
 & \ddots & & & & & & & \\
 & & 1 & & & & & & \\
 & & & c & s & & & & \longleftarrow \text{row } i \\
 & & & -s & c & & & & \longleftarrow \text{row } j \\
 & & & & & 1 & & & \\
 & & & & & & \ddots & & \\
 & & & & & & & 1 & _{n\times n}
\end{array}$$

   is specified by the two numbers c, s and the indices of the rows they apply to, where $c^2 + s^2 = 1$. In this problem, the first Givens rotation is between rows 1 and 2, so the Givens rotation could be stored in compact form as a row vector `[`c`, `s`, 1, 2]`. The entire Q is then stored as a stack of $n-1$ such row vectors. To check your answers, you would need to write a function to multiply out these Givens rotations to form the explicit Q to compare with the output from Matlab's built-in `qr` function.

   You can also use 2×2 Householder reflections if you prefer (you can get the inverse of the product of such H's by simply reversing the order – this can save some bookkeeping)

$$H_{ij}(c,s) = \begin{array}{cccccccc|l}
 & & & \overset{\text{col } i}{\downarrow} & \overset{\text{col } j}{\downarrow} & & & & \\
1 & & & & & & & & \\
 & \ddots & & & & & & & \\
 & & 1 & & & & & & \\
 & & & c & s & & & & \longleftarrow \text{row } i \\
 & & & s & -c & & & & \longleftarrow \text{row } j \\
 & & & & & 1 & & & \\
 & & & & & & \ddots & & \\
 & & & & & & & 1 & _{n\times n}
\end{array}$$