CS 6210 Assignment 3  Due: 10/9/15 (Fri) at 11pm

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. If part of your solution to a problem is based on something found on the Web or in the published literature, then include a citation comment.

Topics: Banded, Block Structured, and Sparse Linear Systems

1  \( A = LDL^H \) for Hermitian Tridiagonal \( A \)

If \( A \in \mathbb{C}^{m \times n} \) and \( B = A^H \in \mathbb{R}^{n \times m} \), then \( b_j \) is the complex conjugate of \( a_{ij} \). If \( A \in \mathbb{C}^{n \times n} \) and \( A = A^H \), then \( A \) is hermitian. If \( A \in \mathbb{C}^{n \times n} \) satisfies \( x^H A x > 0 \) for all nonzero \( x \in \mathbb{C}^n \), then \( A \) is positive definite. If \( A \in \mathbb{C}^{n \times n} \) has the property that \( a_{ij} = 0 \) whenever \( |i - j| > 1 \), then \( A \) is tridiagonal. In MATLAB, if \( A \) is a complex matrix and \( B = A' \), then \( B = A^H \). Complete the following MATLAB function so that it performs as specified.

```matlab
function [d,c,e] = HermFactor(a,g,h)
    % a is a real column n-vector.
    % g and h are real column (n-1)-vectors.
    % Let A be the n-by-n Hermitian tridiagonal matrix with diagonal a and subdiagonal
    % c + i*e, where i^2 = -1. Assume that A is positive definite so that it has
    % the factorization A = L*D*L' where D is diagonal and L is unit lower bidiagonal.
    % d is a real column n-vector so D = diag(d).
    % c and e are real column (n-1)-vectors so that the subdiagonal
    % of L is c + i*e.
    %
    % For full credit, your implementation must not generate and complex vectors or scalars. In otherwords, d, c, and e must be generated using real operations on a, g and h. A script P1 to get you started is available on the course website. Submit HermFactor to CMS.
```

2  Diagonal of the Cholesky Factor

Complete the following function so that it performs as specified:

```matlab
function d = DiagChol(u,tau)
    % Suppose u is a column n-vector, tau>0, and
    % I + tau*u*u' = G*G' is the Cholesky factorization of I + tau*u*u'.
    % d is a column n-vector with d = diag(G), i.e., d(i) = G(i,i), i=1:n
    %
    n = length(u);
    d = diag(chol(eye(n,n)+tau*u*u','lower'));
    % Here is an \( O(n^3) \) implementation with \( O(n^2) \) storage:
    %
    d = diag(chol(eye(n,n)+tau*u*u', 'lower'));
    % It is possible to do much better than that. Indeed, your implementation should involve \( O(n) \) storage and \( O(n) \) work. Hint. Develop recipes for \( d_1, v \in \mathbb{R}^{n-1} \), and lower triangular \( G_1 \in \mathbb{R}^{(n-1)\times (n-1)} \) in
    %
    \[
    GG^T = \begin{bmatrix} d_1 & 0 \\ v & G_1 \end{bmatrix} \begin{bmatrix} d_1 & v^T \\ 0 & G_1^T \end{bmatrix} = I + \tau uu^T
    \]
    % where \( \tau > 0 \). Submit your implementation of DiagChol to CMS.
```
3 SOR and SSOR

The Poisson problem in two dimensions involves finding a function \( u(x, y) \) that satisfies

\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0
\]

on a given region \( R \) subject to the constraint that the value of \( u \) is specified on the boundary of \( R \). Standard discretization of this problem leads to a highly structured but sparse symmetric positive definite linear system \( Au = f \). MATLAB has a pair of built-in functions that can be used to handily set up the matrix \( A \) in sparse format, e.g.,

\begin{verbatim}
g = 20; Region = 'A'; G = numgrid(Region,g); A = delsqr(G)
\end{verbatim}

The first argument to \texttt{numgrid} specifies the region over which the Poisson problem is to be solved. In this case, \textquote{A'} designates a “donut shaped” subset of the unit square. Type \texttt{help numgrid'} to see other options.

The gridsize \( g \) says that a \( g \)-by-\( g \) mesh of points is to be distributed over the unit square and that we will seek to approximate \( u(x, y) \) at those meshpoints which are inside the region \( R \). Download and run the script \texttt{ShowSSOR} to build intuition about the grid \( G \) and the associated matrix \( A \).

The Gauss-Seidel method can be used to solve the linear system \( Au = f \). In particular, if

\[
L = \text{tril}(A,-1) \quad D = \text{diag}(	ext{diag}(A))
\]

then the Gauss-Seidel update is given by

\[
M u^{(k+1)} = N u^{(k)} + f
\]

where \( M = D + L \) and \( N = -L^T \). The error goes to zero like \( \rho(M^{-1}N)^k \) where \( \rho(\cdot) \) is the spectral radius. Unfortunately for Gauss-Seidel, the spectral radius is close to 1 and so convergence is very slow.

The method of successive over-relaxation (SOR) addresses this situation by introducing a parameter \( \omega \) whose value can be used to “dial down” the spectral radius. In SOR we use the splitting \( A = M_\omega - N_\omega \) where

\[
M_\omega = \frac{1}{\omega} D + L \quad N_\omega = \left( \frac{1}{\omega} - 1 \right) D - L^T \quad 1 \leq \omega \leq 2
\]

Notice that \( \omega = 1 \) corresponds to Gauss-Seidel. As \( \omega \) increases, we effectively move more and more of \( D \) from \( M_\omega \) to \( N_\omega \). The error in SOR goes to zero like \( \rho(M_\omega^{-1} N_\omega)^k \).

The symmetric SOR method (discussed in GVL4 §11.2.7) generates \( x^{(k)} \) from \( x^{(k-1)} \) by solving

\[
M_\omega y^{(k)} = N_\omega x^{(k-1)} + b
\]

for \( y^{(k)} \) and then solving

\[
M_\omega^T x^{(k)} = N_\omega^T y^{(k)} + b
\]

for \( x^{(k)} \). The error in SSOR goes to zero like \( \rho(M_\omega^{-T} N_\omega^T M_\omega^{-1} N_\omega)^k \).

Modify the \texttt{ShowSSOR} subfunction \texttt{AnalyzeThis} so that it graphically displays the effect that \( \omega \) has on the spectral radius \( \rho(M_\omega^{-1} N_\omega) \) and the spectral radius \( \rho(M_\omega^{-T} N_\omega^T M_\omega^{-1} N_\omega) \). In particular, instead of displaying a spy plot of the matrix \( A \) in \texttt{subplot(1,2,2)}, \texttt{AnalyzeThis} should display (in one plot window) graphs of \( \rho(M_\omega^{-1} N_\omega) \) and \( \rho(M_\omega^{-T} N_\omega^T M_\omega^{-1} N_\omega) \) that are based on evaluating these functions at \( \omega = 1:\text{inspace}(1,2,20) \). The spectral radius of a sparse matrix \( Z \) can be (inefficiently) computed via \texttt{max(abs(eig(full(Z))))}. You can experiment with different grid sizes, but since the dimension of \( A \) grows as the square of this parameter, you don’t want make it much larger than 25 or so. Submit your updated version of \texttt{ShowSSOR} to CMS.
4 Incomplete $LDL^T$ and $LDM^T$

For background, read about the incomplete Cholesky factorization idea in GVL4 §11.5.8. You are going to implement the Lin-More strategy to produce incomplete versions of these factorizations

$$A = LDL^T \quad L \text{ unit lower triangular, } D = \text{diag}(d_1, \ldots, d_n)$$  \hfill (1)

$$A = LDM^T \quad L, M \text{ unit lower triangular, } D = \text{diag}(d_1, \ldots, d_n)$$  \hfill (2)

In (1), we assume $A$ is symmetric and positive definite. Note that $LD^{1/2}$ is the Cholesky factor. We have seen LDL in the context of symmetric positive definite tridiagonal systems. In (2) we will assume that $A$ is diagonally dominant so that the no-pivot strategy is stable. Note that $A = L(DM^T)$ is the traditional LU factorization. The reason to prefer LDM over LU in this problem is that the normalization of the lower and upper triangular parts is the same. (In LU, $L$ is unit lower triangular and $U$ is not.)

Gaxpy implementations that can be used to compute these factorizations are available via ShowLDL_LDM.m. Here they are:

```matlab
function [L,d] = LDL(A)
    % A is an nxn and symmetric positive definite matrix
    % L is unit lower triangular and d is a column n-vector such that
    % A = LDL' where D = diag(d).
    [n,n] = size(A);
    d = zeros(n,1); L = eye(n,n);
    for j=1:n
        s = d(1:j-1).*L(j,1:j-1)';
        d(j) = A(j,j) - L(j,1:j-1)*s;
        v(j+1:n) = A(j+1:n,j) - L(j+1:n,1:j-1)*s;
        L(j+1:n,j) = v(j+1:n)/d(j);
    end
end
```

```matlab
function [L,d,M] = LDM(A)
    % A is nxn and has an LDM factorization
    % L and M are nxn and unit lower triangular and d is a column n vector
    % such that A = LDM' where D = diag(d).
    [n,n] = size(A);
    d = zeros(n,1); L = eye(n,n); M = eye(n,n);
    for j=1:n
        s = d(1:j-1).*L(j,1:j-1)';
        d(j) = A(j,j) - L(j,1:j-1)*s;
        v(j+1:n) = A(j+1:n,j) - L(j+1:n,1:j-1)*s;
        L(j+1:n,j) = v(j+1:n)/d(j);
        t = d(1:j-1).*L(j,1:j-1)';
        w(j+1:n) = A(j,j+1:n) - t'*M(j+1:n,1:j-1)';
        M(j+1:n,j) = w(j+1:n)/d(j);
    end
end
```

Implement a function $[L,d] = LDLinc(A,p)$ that returns an approximate LDL factorization defined by the nonnegative integer $p$. In particular, you must modify $v(j+1:n)$ before it is used to compute $L(j+1:n,j)$:

Set to zero each component of $v(j+1:n)$ that is not one of the $\text{nnz}(A(j+1:n,j)) + p$ largest entries in $|v(j+1:n)|$.

Here “nnz” means “number of nonzeros.” This strategy ensures that the number of nonzeros in $L$ is bounded:

$$\text{nnz}(L) \leq \text{nnz}(\text{tril}(A,-1)) + np.$$  

Also implement a function $[L,d,M] = LDMinc(A,p,q)$ that returns an approximate LDM factorization defined by the nonnegative integers $p$ and $q$. In particular, you must modify $v(j+1:n)$ and $w(j+1:n)$ before they are used to compute $L(j+1:n,j)$ and $M(j+1:n,j)$:
Set to zero each component of $v(j+1:n)$ that is not one of the $\text{nnz}(A(j+1:n,j)) + p$ largest entries in $|v(j+1:n)|$.
Set to zero each component of $w(j+1:n)$ that is not one of the $\text{nnz}(A(j,j+1:n)) + q$ largest entries in $|w(j+1:n)|$.

This ensures that the number of nonzeros in $L$ and $M$ is bounded:

$$\text{nnz}(L) \leq \text{nnz}(\text{tril}(A,-1)) + np$$
$$\text{nnz}(U) \leq \text{nnz}(\text{triu}(A,1)) + nq$$

Additional comments and requirements:

- If $n - j \leq p + \text{nnz}(A(j+1:n,j))$, then we are “allowed” to have a full $L(j+1:n,j)$.
- If $n - j \leq q + \text{nnz}(A(j,j+1:n))$, then we are “allowed” to have a full $M(j+1:n,j)$.
- The matrices $L$ and $M$ must be in sparse format.
- Think hard about your manipulations with the $v$ and $w$ vectors. Are they efficient?

Submit $\text{LDLinc}$ and $\text{LDMinc}$ to CMS. A pair of test scripts $\text{P4A}$ and $\text{P4B}$ are available on the course website.