CS4220 Assignment 4 Due: 3/14/13 (Thur) at 11pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the solutions you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group. Each problem is worth 5 points. One point may be deducted for poor style.

Topics: Sparse Cholesky, Least Squares, Gauss-Seidel, Jacobi, Givens Rotations

1 Out-of-Core Least Squares Using the QR Factorizartion

Review the Givens QR factorization algorithm by running the script `Mar1Lec` that is available off of the syllabus page. In this problem you are going to reorder the zeroing process in a way that makes the process more friendly for large least square problems that have many more rows than columns.

The starting point is to show how to compute the QR factorization of a matrix that looks like this:

$$C = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ \times & \times & \times & \times \end{bmatrix}$$

Four Givens rotations do the job...

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{(1,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{(2,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{(3,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Let's call this the *C update process.* The index pairs specify what the Givens rotation looks like. For example, here is a closer look at the "(2,5) transition":

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & s \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -s & 0 & 0 & c \end{bmatrix}^T \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & \times & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

Draw pictures of the other transitions so that you see what is going on. Think about general n (instead of $n = 4$. Think about how to make each transition update efficient. (Only 2 rows are involved and they each have a bunch of zeros.)

Now suppose we want to minimize $\| Ax - b \|_2$ where $A \in \mathbb{R}^{m\times n}$ is large and sparse. Here is the plan assuming that C is an $(n+1)$-by-n array:

```
Copy $A(1{:}n+1, :)$ into $C$.
By any means, compute the QR factorization of $C$ and update $b$ accordingly.
**for** $i = n+2{:}m$
    Copy $A(i, :)$ into $C(n+1, :)$.
    Execute the $C$ update process making sure to apply the Givens rotations to $b$.
**end**
Solve the upper triangular system $C(1{:}n, 1{:}n)x_{LS} = b(1{:}n)$
```

Here is the story behind this strategy. A is so large that we cannot store it all in "fast memory" even if it is represented in sparse format. However, we do have room to store an $(n+1)$-by-n matrix in full format. So after an initialization, we "read in" A one row at a time updating the QR factorization as we go along.

Write a MATLAB function `xLS = OutOfCoreLS(A,b)` that implements this strategy. Assume that `A` is an m-by-n matrix with full column rank and represented in sparse format. Assume that `b` is m-by-1. Regarding the various "copy" operations in the above pseudocode, remember that MATLAB handles the format conversion in assignment statements of the form `Full Format Matrix` $\leftarrow$ `Sparse Format Matrix`. A tricky part of the problem concerns the proper update of b. In this regard remember two things. First, the Givens update $b \leftarrow G^T b$ must accompany the Givens update $A \leftarrow G^T A$. Second, updates of the form $A \leftarrow G^T A$ are carried out on C so be clear about just what rows of A are housed in C.

A test script `P1` is available on the course website. Submit your implementation to CMS.

2 Block Gauss-Seidel

We wish to solve the block system

$$\begin{bmatrix} A_1 & C \\ C^T & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where it is assumed that (a) the matrix of coefficients

$$A = \begin{bmatrix} A_1 & C \\ C^T & A_2 \end{bmatrix}$$

is symmetric, positive definite and sparse, (b) $A_1 \in \mathbb{R}^{n_1 \times n_1}$ and $A_2 \in \mathbb{R}^{n_2 \times n_2}$ are both large and sparse and have nicely sparse Cholesky factors, and (c) the matrix of coefficients does *not* have a nicely sparse Cholesky factor. We propose to solve such a system by implementing a block version of Gauss-Seidel:

```
Produce initial guess $x_1^{(0)}$ and $x_2^{(0)}$.
$k = 1$
**while** $k < itMax$ and the relative residual is bigger than $tol$
    Solve $A_1 x_1^{(k)} = b_1 - C x_2^{(k-1)}$
    Solve $A_2 x_2^{(k)} = b_2 - C^T x_1^{(k)}$
    $k = k + 1$
**end**
```

The relative residual is given by

$$\frac{\| Ax^{(k)} - b \|_2}{\| b \|_2}$$

where A is the matrix of coefficients,

$$x^{(k)} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

and *itMax* is an iteration maximum.

The system solves that involve A_1 and A_2 should be done using the sparse version of MATLAB's `chol` function:

```
[G,p,S] = chol(A,'lower')
```

Here, `A` is symmetric, positive definite and in sparse format. It computes lower triangular `G` in sparse format so that

$$S^T A S = G G^T$$

where `S` is a permutation in sparse format. Develop a function

```
[x1,x2,its] = SparseSolve(A1,A2,C,b1,b2,tol,itMax)
```

that implements this procedure. It should return in `x1` and `x2` the vectors $x_1^{(k)}$ and $x_2^{(k)}$ and the value of k in `its`. If $k < itMax$ then the relative residual should be less than *tol*. Otherwise, `x1` returns $x_1^{(itMax)}$ and `x2` returns $x_2^{(itMax)}$. You are free to exploit the fact that MATLAB "does the right thing" in operations like $y = G \backslash b$ and and $z = S * y$.

A test script `P2` is available on the course website together with files `A1.tex`, `A2.tex`, and `C` that house the sparse matrices A_1, A_2, and C. Submit your implementation of `SparseSolve` to CMS.

The test script example is nice in two ways. First, we note that since

$$\| b \| = \| Ax \| \leq \| A \| \| x \|$$

then

$$\| x^{(k)} - x \| \;=\; \| A^{-1}(Ax^{(k)} - b) \| \;\leq\; \| A^{-1} \| \frac{\| Ax^{(k)} - b \|}{\| b \|} \| b \| \;\leq\; \| A^{-1} \| \| A \| \frac{\| Ax^{(k)} - b \|}{\| b \|} \| x \|$$

Thus, if the relative residual is less than *tol* then the relative error in $x^{(k)}$ is less than cond$(A)\cdot tol$. The test problem condition is small so the method produces an accurate x. Second, convergence of the method depends $|\lambda|^k$ where λ is largest eigenvalue of $A_2^{-1}C^TA_1^{-1}C$ in absolute value. (Why?) This quantity can be horribly close to one, but in our problem it is not.

3 Polynomial Data Fitting

Read about polynomial data fitting in §6.1 in AG. Let $f(t)$ be the function

$$f(t) \;=\; .05\sin(1000t) \;+\; .5\cos(\pi t) \;-\; .4\sin(10t).$$

Let $p_k(t)$ be the degree k polynomial that minimizes

$$\phi(p) \;=\; \sum_{i=1}^{101} [p(t_i) - f(t_i)]^2$$

where $t = linspace(0, 1, 101)$. Write a script `ShowLSfit` that produces six plots each displayed in its own figure window. Plot k should depict both f and p_k. The idea is to filter out the $\sin(1000t)$ term. Details:

- Base the plot of f on `linspace(0,1,500)` and make sure you can see the fitting polynomial clearly.
- Solve the LS problems using the QR factorization. Even though there are six least squares problems to solve, you only need one QR factorization. Hint. If `[Q,R] = qr(A,0)`, then $Q(:, 1{:}k)R(1{:}k, 1{:}k)$ is the thin QR factorization of $A(:, 1{:}1k)$

Submit `ShowLSfit` to CMS.