CS 475/675 Spring 2015: Assignment 1

Due June 1, 2015, 5:00pm.
Written/analytical work (i.e., Q.1-5) should be submitted to the physical assignment box on 4th floor MC.
MATLAB code and any associated output and commentary should be submitted to the DropBox on LEARN. (i.e., Q.6-7)
CS475 should do questions 1-6. CS675 should do *all* questions.

For full marks, be sure to show all your work!

1. **(10 marks)** *Column-oriented backward solve.* Consider solving a 3×3 system $Ux = b$ where U is upper triangular. One approach to its solution is the following process. First, resolve x_3. It can then be removed from equations 1 and 2, and we can proceed with the reduced system in which the right-hand-side has been updated. We next compute x_2, remove it from equation 1, etc. For example, if this approach is applied to

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 2 \\ 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 12 \end{bmatrix} \tag{1}$$

   we first compute $x_3 = 2$, and then solve the reduced 2×2 system:

$$\begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \end{bmatrix} - 2 \begin{bmatrix} 3 \\ 2 \end{bmatrix} \tag{2}$$

   Derive an efficient algorithm for the general case where the matrix U is an $n \times n$ upper triangular matrix (not necessarily unit diagonal). Present your algorithm in pseudocode. The inputs to your algorithm should be the matrix U and the right-hand side b, and the output should be the solution x. What is the complexity of your algorithm? Show your work.

2. **(10 marks)** Consider an $n \times n$ matrix A with upper bandwidth q and lower bandwidth p. Suppose you are given the L and U matrices comprising the LU decomposition of A, i.e., $A = LU$. Derive an *efficient* algorithm (i.e., forward and backward solves) to solve $Ax = b$. Give precise pseudocode for your algorithm. What is the complexity of your algorithm? Show your work.

3. **(10 marks)** Given an $n \times n$ matrix A and its LU factors, how do you *efficiently* solve the following problems using the given matrices L and U. Note that you should not need to compute the explicit inverse of A or any other matrix.

   (a) Compute $\alpha = c^T A^{-1} b$.

   (b) Solve the matrix equation $Bx = b$ where $B = A + \tilde{A}$. The matrix $\tilde{A}$ satisfies the condition that all of its columns are zero vectors, except the kth column where $\tilde{A}_{i,k} = i$, for $1 \leq i \leq n$. (Hint: Consider applying the Sherman-Morrison formula.)

4. **(5 marks)** Some finite difference approximations of the Poisson equation, $\Delta\phi = f$, can give rise to a *9-point stencil* involving the current point and its 8 surrounding immediate neighbours. This results in a set of linear equations of the general form:

$$\theta T_{i-1,j-1} + \gamma T_{i,j-1} + \mu T_{i+1,j-1} + \rho T_{i-1,j} + \eta T_{i,j} + \delta T_{i+1,j} + \nu T_{i-1,j+1} + \alpha T_{i,j+1} + \beta T_{i+1,j+1} = f_{i,j}$$

for $i = 1, 2, ...m$. Let A be the matrix of the linear system. Assuming the natural ordering of the unknowns, draw a picture of A. Describe the sparsity structure of A and indicate which coefficients (Greek letters) appear in which non-zero entries.

5. **(10 marks)** Consider a *symmetric* matrix whose graph is given by:

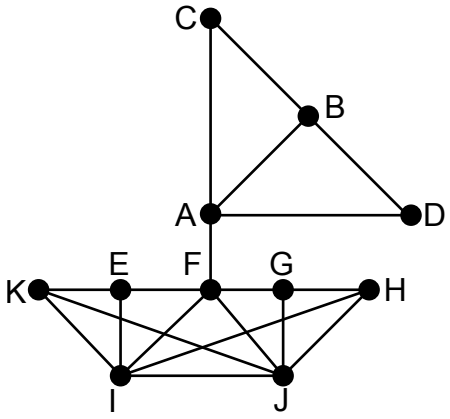

Figure 1: Awesome Boat / Matrix Graph

Perform Cuthill-McKee (starting with node A), reverse Cuthill-McKee, and minimum degree re-orderings on this matrix. For the minimum degree ordering, show the graph at each stage of elimination. Ties should be broken by selecting the node with the alphabetically earlier original label (as given on the graph). In each case, redraw the graph using the new labels.

6. **(40 marks)** In modeling (steady state) heat flow, the temperature $T = T(x, y)$ satisfies the Poisson equation

$$-\frac{\partial^2 T}{\partial x^2} - \frac{\partial^2 T}{\partial y^2} = f(x, y)$$

where $f(x, y)$ represents the heat source function.

We approximate $T(x, y)$ at discrete locations on a two dimensional grid with m active grid points in each dimension. Let the (i, j) grid point have location (x_i, y_j) where $x_i = ih$, $y_i = jh$, $h = 1/(m + 1)$ is the grid spacing, and $i, j \in [0, m + 1]$. If we let $T_{i,j} \approx T(x_i, y_j)$, then the finite difference approximation results in a set of linear equations

$$\frac{1}{h^2}(4T_{i,j} - T_{i-1,j} - T_{i+1,j} - T_{i,j-1} - T_{i,j+1}) = f_{i,j}. \tag{3}$$

We will assume that all the boundary temperatures along the sides of the grid are zero:

$$T_{0,j} = T_{i,0} = T_{m+1,j} = T_{i,m+1} = 0$$

for all $i, j \in [0, m + 1]$. We want to analyze the heat flow with two heat sources as given by:

$$f_{i,j} = \begin{cases} 1 \text{ if } \|(x_i, y_j) - (0.35, 0.6)\| \leq 0.1 \\ 1 \text{ if } \|(x_i, y_j) - (0.8, 0.25)\| \leq 0.1 \\ 0 \text{ otherwise.} \end{cases}$$

An example of a final temperature distribution for a different set of three heat sources is shown in Figure 2.

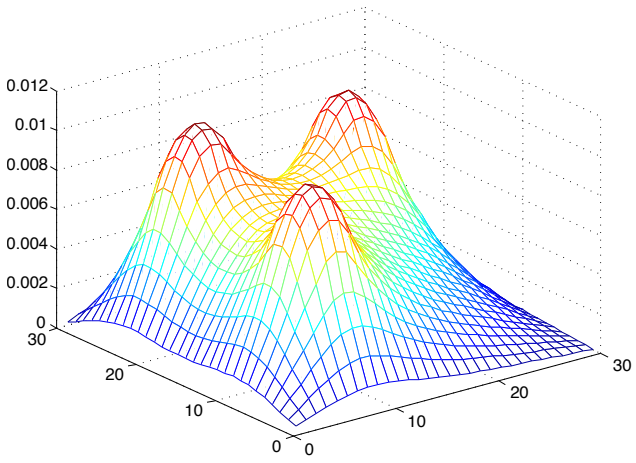

Figure 2: Example plot for three heat sources

Define a vector x such that

$$x_k = T_{i,j},$$

where $k = (j-1)m + i$. Similarly, we define the vector b with $b_k = f_{i,j}$. Then we can write equation (3) in matrix form

$$Ax = b. \tag{4}$$

The coefficient matrix A is sparse with size $n \times n$, where $n = m^2$.

(a) Create a MATLAB function `[A,b] = Lap2D(m)`. The input is a positive integer m, the number of active grid points in each dimension. The outputs are the matrix A and the right-hand side b as in equation (4).

(b) Implement the following numerical methods: Gaussian elimination, Cholesky factorization, and Band Gaussian elimination. Create the following MATLAB functions:

```
x=GaussElim(A,b)
x=Cholesky(A,b)
x=BandGE(A,b,p,q)
```

These MATLAB functions take as inputs the matrix A and right-hand side b and compute the solution x using the corresponding variant of Gaussian elimination. The parameters `p` and `q` indicate the lower and upper matrix bandwidth, respectively. (There is no need to implement pivoting or check for zero or small pivots.) You can check your code by visualizing the solution x. Use the following command:

```
>> mesh(reshape(x,m,m))
```

This will convert the solution vector x into a 2D array, and generate a 2D mesh plot of the result. Submit a 2D mesh plot of the solution for the case when $m = 20$.

(c) Create a MATLAB script, `GETimes.m`, that solves (4) using the three different variants of Gaussian elimination you wrote. In particular, set up the matrix A and right-hand side b by calling the MATLAB function `Lap2D`. Then solve the equation by calling one of the MATLAB functions in part (b). Record the CPU time using the MATLAB command `cputime`. Construct a table of execution times for solving with each of the

methods above. Try the grid sizes $8\times 8, 16\times 16, 24\times 24, 32\times 32$. (Consider vectorizing the innermost loop of your implementations to keep solve times more manageable.) Compare and comment on the timing results for the three methods you implemented.

Submit to the LEARN Dropbox: `Lap2D.m`, `GaussElim.m`, `Cholesky.m`, `BandGE.m`, `GETimes.m`, a mesh plot of x, a table of CPU times, and your comments on the timing results.

7. [**CS675 students only**] **(15 marks)** Consider using finite differences to discretize the three-dimensional version of the heat equation in question 6.

   (a) Create a MATLAB function:

   ```
   A = Lap3D(m)
   ```

   The input is m, the number of active grid points in each dimension. The output is the sparse matrix A, whose size is $n \times n$ where now we have $n = m^3$. You do not need to build b.

   (b) Create another MATLAB function:

   ```
   B = ReorderedLap3D(A,ordering_method)
   ```

   that takes your Laplacian matrix and produces a reordered version. The first input parameter is the Laplacian matrix A produced by `Lap3D`. The second input, `ordering_method`, is a string that can be one of the following: 'natural', 'rcm', 'min', which correspond to the natural, reverse Cuthill-McKee (RCM), and minimum degree orderings, respectively. You will use MATLAB's reordering functionality to generate reordered versions of the matrix A. For example, to construct the ordering vector for RCM and then apply it to the matrix, do the following:

   ```
   >> p = symrcm(A);
   >> B = A(p,p);
   ```

   Then the matrix B is the reordering of A using the RCM ordering. The corresponding MATLAB command for the minimum degree ordering is `symamd`.

   (c) Create a MATLAB script, `CompareNNZ.m`, that compares the number of nonzeros in the Cholesky factorization of A using different orderings. The program should call `Lap3D` to create a sparse matrix A, and then use `ReorderLap3D` to perform each of the reorderings. Then compute the Cholesky factor G using the MATLAB command `chol`. Determine the number of nonzeros of G using the MATLAB command `nnz`. Construct a table of number of nonzeros in A and G for different ordering methods and grid sizes. Use grid resolutions of $m = 8, 16, 24, 32, 40$. Comment on the number of nonzeros in different cases. In particular, how are the number of nonzeros affected by the ordering methods and grid dimensions? Which ordering method produces the fewest nonzeros in G? Produce a sparsity plot of G for each of the three ordering methods for the $m = 24$ case, using MATLAB's `spy` command.

Submit to the LEARN Dropbox: `Lap3D.m`, `ReorderLap3D.m`, `CompareNNZ.m`, the three sparsity plots, a table of numbers of nonzeros, and your comments on the results.