MAT 226B, Winter 2018

Homework 2

(due by Tuesday, February 6, 11:59 pm)

General Instructions

- You are required to submit your homework by uploading a single pdf file to Canvas. Note that the due dates set in Canvas are hard deadlines. I will not accept any submissions outside of Canvas or after the deadline.
- If at all possible, use a text processing tool (such as LaTeX) for the preparation of your homework. If you submit scanned-in hand-written assignments, make sure that you write clearly and that you present your solutions in a well-organized fashion. If I cannot read your homework, I will not be able to grade it!
- Feel free to discuss the problems on the homework sets with other students, but you do need to submit your own write-up of the solutions and your own MATLAB codes. If there are students with solutions that were obviously copied, then each involved student (regardless of who copied from whom) will only get the fraction of the points corresponding to the number of involved students.
- Test cases for computational problems are often provided as binary Matlab files. For example, suppose the file "`LS.mat`" contains the coefficient matrix A and the right-hand side b of a system of linear equations. The Matlab command "`load('LS.mat')`" will load A and b into Matlab.
- When you are asked to print out numerical results, print real numbers in 15-digit floating-point format. You can use the Matlab command "`format long e`" to switch to that format from Matlab's default format. For example, the number 10π would be printed out as `3.141592653589793e+01` in 15-digit floating-point format
- When you are asked to write Matlab programs, include printouts of your codes in your homework.

---

1. Matlab provides a Cholesky routine "`chol`" for matrices $A \succ 0$. The routine exploits sparsity when applied to a sparse matrix A. The Matlab call

   ```
   L = chol(A,'lower');
   ```

   computes the Cholesky factor L of A.

Matlab also provides the routines "`symamd`", "`colamd`", "`symrcm`", and "`colperm`" that implement heuristics for performing the symbolic factorization step needed in sparse Cholesky factorization. The output of these routines is a compact representation of the permutation matrix P in the sparse Cholesky factorization

$$P^T AP = LL^T.$$

The use of this compact representation (in the case of "`symamd`") is as follows:

```
p = symamd(A);
L = chol(A(p,p),'lower');
```

The Matlab functions in "`make_2d_laplacian.m`" and "`make_3d_laplacian.m`" generate sparse matrices $A \succ 0$ that correspond to discretizations of $-\Delta$ on equidistant rectangular grids in two and three dimensions, respectively.

(a) The Matlab call

```
A = make_2d_laplacian(m);
```

produces an $n \times n$ matrix A with $n = m^2$. For $m = m_0 := 67$, compute the Cholesky factor L of A for the following 5 cases:

(i) No reordering of the rows and columns of A;

(ii) Reordering with `symamd`;

(iii) Reordering with `colamd`;

(iv) Reordering with `symrcm`;

(v) Reordering with `colperm`.

For all 5 runs, report the number of nonzero entries of the Cholesky factor L and submit a plot of the sparsity structure of L (as produced by the Matlab command `spy(L)`).

For $m = 2^i m_0$, $i = 0, 1, \ldots$, compute the Cholesky factor L of A using the `symamd` ordering. As i increases, you will run out of storage or CPU time. For which i does this happen on the machine you run Matlab on?

(b) The Matlab call

```
A = make_3d_laplacian(m);
```

produces an $n \times n$ matrix A with $n = m^3$. For $m = m_0 := 37$, compute the Cholesky factor L of A for the cases (i)–(v) listed in part (a).

For all 5 runs, report the number of nonzero entries of the Cholesky factor L and submit a plot of the sparsity structure of L (as produced by the Matlab command "`spy(L)`").

For $m = 2^i m_0$, $i = 0, 1, \ldots$, compute the Cholesky factor L of A using the `symamd` ordering. As i increases, you will run out of storage or CPU time. For which i does this happen on the machine you run Matlab on?

2. You are given an 9×9 matrix $A \succ 0$ with the following sparsity structure:

$$A = \begin{bmatrix} * & 0 & 0 & * & 0 & 0 & 0 & 0 & * \\ 0 & * & 0 & * & * & * & * & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 & * & 0 \\ * & * & 0 & * & 0 & * & 0 & 0 & 0 \\ 0 & * & 0 & 0 & * & 0 & 0 & * & 0 \\ 0 & * & 0 & * & 0 & * & * & 0 & * \\ 0 & * & 0 & 0 & 0 & * & * & * & 0 \\ 0 & 0 & * & 0 & * & 0 & * & * & 0 \\ * & 0 & 0 & 0 & 0 & * & 0 & 0 & * \end{bmatrix}.$$

   (a) Show the graph associated with A.

   (b) Use the minimum degree algorithm to reorder the matrix. (Break ties by giving priority to the node with lowest index.)

   (c) Show the sparsity structure of the sparse Cholesky factor L of $P^T AP$, where P is the permutation matrix corresponding to the minimum-degree ordering determined in part (b).

3. Suppose that at the beginning of the k-th step of sparse LU factorization, the matrix $U^{(k)}$ has the following sparsity structure:

$$U^{(k)} = \big[u_{il}\big] = \begin{bmatrix} * & * & * & 0 & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & * & * & 0 & 0 & 0 & * & * & 0 \\ * & 0 & 0 & 0 & 0 & * & 0 & 0 & * & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & 0 & * & * \\ * & 0 & * & * & 0 & * & 0 & 0 & 0 & * & 0 \\ 0 & * & 0 & 0 & 0 & * & 0 & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & 0 & * & 0 \\ * & 0 & 0 & * & 0 & 0 & 0 & * & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & * & 0 & 0 & 0 & 0 & * \end{bmatrix}.$$

   Moreover, the nonzero entries $u_{il} \neq 0$ of $U^{(k)}$ are assumed to satisfy

$$\min_{i,l\,:\,u_{il}\neq 0} \big|u_{il}\big| \;\geq\; \frac{1}{3} \max_{i,l} \big|u_{il}\big|.$$

   (a) Determine the number of fill-in elements and their location if the k-th step is performed without pivoting.

   (b) Determine the number of fill-in elements and their location if the k-th step is performed with a pivot element determined via the practical Markowitz criterion (with parameter $\alpha = 0.1$) presented in class.

4. Recall that in compressed sparse column format, sparse matrices $A = \big[a_{jk}\big] \in \mathbb{R}^{n\times n}$ are stored as two integer vectors J and I and a real vector V_A containing the nonzero entries $a_{jk} \neq 0$ of A. More precisely, J contains the row indices of the nonzero entries listed column by column, and I contains pointers to the columns. The indices (j,k) of the nonzero entries are then given by

$$\big(J(i),k\big), \quad i = I(k), I(k)+1, \ldots, I(k+1)-1, \quad k = 1,2,\ldots,n.$$

(a) Let $L \in \mathbb{R}^{n\times n}$ be a unit lower-triangular matrix given in Matlab's sparse matrix format. Employ Matlab's "`find`" command to write an efficient Matlab function that generates integer vectors J and I and a real vector V_L that store the strictly lower-triangular part of L in compressed sparse column format.

To test your function, first apply it to the matrix L provided in "`small_ex.mat`" and print out J, I, and V_L.

Then apply your function to the matrix L provided in "`large_ex.mat`" and print out

$$I(50000),\ I(100000),\ I(150000),\ I(200000),\ I(250000).$$

Hint: Recall that internally, Matlab stores sparse matrices in compressed sparse column format. For a triangular matrix, one can use "`find(J-K == 0)`" to efficiently construct the pointer vector I. Here, J and K are the vectors of row and column indices produced by "`find(L)`". Note that the Matlab command "`tril(L,-1)`" returns the strictly lower-triangular part of L.

(b) Let $U \in \mathbb{R}^{n\times n}$ be a nonsingular upper-triangular matrix given in Matlab's sparse matrix format. Employ Matlab's "`find`" command to write an efficient Matlab function that generates integer vectors J and I and a real vector V_U that store U in compressed sparse column format.

To test your function, first apply it to the matrix U provided in "`small_ex.mat`" and print out J, I, and V_U.

Then apply your function to the matrix U provided in "`large_ex.mat`" and print out

$$I(50000),\ I(100000),\ I(150000),\ I(200000),\ I(250000).$$

(c) Write two Matlab functions that solve systems of linear equations with unit lower-triangular coefficient matrices L and nonsingular upper-triangular coefficient matrices U, respectively. Here, L and U are given in the compressed sparse column format produced as output of your Matlab functions from (a) and (b).

To test your functions, first use them to solve the two triangular systems

$$Lc = b \quad \text{and} \quad Ux = c, \tag{1}$$

where L, U, and b are provided in "`small_ex.mat`", and print out x.

Then use your functions to solve the systems (1), where L, U, and b are provided in "`large_ex.mat`". Print out the entries

$$x(50000),\ x(100000),\ x(150000),\ x(200000),\ x(250000)$$

of x.

5. Write a Matlab program that uses Matlab's LU factorization "`lu(A,'vector')`" and your functions for triangular solves with L and U from Problem 4(c) to compute the solution of

$$Ax = b, \tag{2}$$

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular sparse matrix and $b \in \mathbb{R}^n$. Set up your program so that you can run "`lu`" both with scaling D and without scaling. Also allow for column reordering of A, i.e., "`lu`" is applied to $A(:, p_0)$, where p_0 is a suitable reordering of the columns.

To test your program, use it to solve the linear system (2) for the two cases provided in "`large_ex1.mat`" and "`large_ex2.mat`". Run your program with and without scaling, and

(i) without column reordering,

(ii) with column reordering given by $p_0 = \texttt{colamd}(A)$,

(ii) with column reordering given by $p_0 = \texttt{colperm}(A)$,

for a total of 6 runs for each of the two systems.

For each of your 12 runs, print out

$$\texttt{nnz}(L), \quad \texttt{nnz}(U), \quad \frac{\|b - Ax\|_2}{\|b\|_2}$$

and the entries

$$x(10),\ x(100),\ x(1000),\ x(100000),\ x(200000)$$

of x.