CS 290H: Sparse matrix algorithms // Homework 3

Assigned April 16, 2008

due April 23

1. [20 points] Let G be the graph of the n-vertex model problem, that is, a k-by-k grid graph with $n = k^2$ vertices. Prove that there is some constant $c > 0$ such that for *every* elimination ordering on G, the filled graph G^+ contains a complete subgraph with at least $c\sqrt{n}$ vertices.

(Hint: Suppose you're given an ordering for the vertices of G. Think of playing the graph game in the given order, and consider the first time that you've either marked all the vertices in any single row of the grid or else marked all the vertices in any single column.)

2. [40 points] Let A be an $n \times n$ symmetric, positive definite matrix, and let $G = G(A)$ be its graph. Fix an elimination ordering of G (that is, a labeling of the vertices of G by the integers 1 through n), and let G^+ be the filled graph for that ordering. We define the *front size* of the ordering as the maximum number of higher-numbered neighbors of any vertex in G^+,

$$s = \max_{1 \le v \le n} \{\#(w) \mid w > v \text{ and } (v, w) \text{ is an edge of } G^+\},$$

which is also equal to the maximum number of nonzeros below the diagonal in any column of the Cholesky factor L of A.

2(a) Using the path lemma, prove that the front size of the n-vertex model problem, ordered row-by-row, is $O(\sqrt{n}\,)$.

2(b) Prove that there's no elimination order for the model problem that gives front size better than $O(\sqrt{n}\,)$—in other words, prove that there is some constant c for which every elimination ordering gives front size at least $c\sqrt{n}\,$. (You may use the result of problem (1) here.)

2(c) The *height* of a tree is the length in edges of the longest path from a leaf to the root. Thus the height of a one-vertex tree is 0. Prove that for any symmetric positive definite matrix A, the front size of A is at most the height of its elimination tree $T(A)$.

2(d) A *planar graph* is one that can be drawn in the plane without any edges crossing. The *planar separator theorem* states that if G is any planar graph with n vertices, then there is subset S of the vertices (called a *separator*) for which

1. removing S from G leaves no connected component with more than $2n/3$ vertices, and
2. S itself has at most $\sqrt{6n}$ vertices.

Using the planar separator theorem in a divide-and-conquer fashion, prove that every n-vertex planar graph has an elimination ordering with front size $O(\sqrt{n}\,)$.

3. **[40 points]** (see Davis problem 6.13). An *incomplete LU factorization* is an approximate factorization $A \approx LU$, in which L and U are lower and upper triangular matrices whose product is "approximately" A in some sense, but L and U have fewer nonzeros than the actual LU factors of A. It is useful as a preconditioner for iterative methods. There are many ways to define and compute a so-called ILU factorization; we'll study some of them in the second part of the course.

For this problem, you are to write a mex-file that implements the following version of ILU. I suggest that you start with Davis's `cs_lu` code, though can write your code from scratch if you prefer. Your ILU routine should take two additional parameters `droptol` and `diagtol`. It computes one column of the factors at a time, just like `cs_lu`, with three differences:

- There is no partial pivoting, that is, no exchanging rows.
- After each column of L and U has been computed (but before the column of L is scaled by the diagonal element u_{jj}), let η be the norm of that computed column. All fill entries in that column that are smaller in magnitude than the "local drop tolerance," which is $\texttt{droptol} \cdot \eta$, are "dropped" from L or U. Notice that only fill entries are dropped—nonzeros of L and U that correspond to nonzeros of the original matrix A are kept no matter how small they are. The other exception to the dropping rule is the diagonal element u_{jj} of U, which is also kept even if it is too small.
- If $|u_{jj}| < \texttt{diagtol} \cdot \eta$, then u_{jj} is replaced by $\pm(\texttt{diagtol})^{1/2} \cdot \eta$. Like the dropping, this replacement happens before the column of L is scaled by u_{jj}.

Your code should also return the number of diagonal elements that were adjusted because of `diagtol`.

Setting `droptol = diagtol = 0` produces the complete factorization $A = LU$ without pivoting, if it exists. Setting `droptol = 1` produces what is sometimes called an "ILU0" factorization, in which all fill elements are dropped.

Verify and experiment with your code on various matrices in Matlab. You can compare your code to Matlab's `luinc()` function, but notice that they don't do quite the same thing. I will post some suggestions of matrices to experiment with as well.

You will use your ILU code for preconditioning experiments in the second part of the course.