CS 290H: Sparse matrix algorithms // Homework 5

Assigned April 30, 2004

Due by class time Monday, May 12

1. [10 points] The object of this problem is to measure experimentally the convergence rate of CG on the (three-dimensional) model problem with various versions of incomplete factorization preconditioning and with various orderings. You can generate the n-by-n matrix of the 3D model problem by

```
A = grid3d(k);
```

where $n = k^3$. (The routine `grid3d.m` is in the `meshpart` subdirectory of the Matlab codes linked from the course web site.) Generate a right-hand side b as a random n-vector.

For each choice of k that you make, experiment with the following four permutations of the matrix A:

- The natural ordering, as generated by `grid3d`.
- The bandwidth-limiting heuristic "reverse Cuthill-McKee," as implemented by Matlab's `symrcm`.
- The approximate minimum degree heuristic, as implemented by Matlab's `symamd`.
- A "red-black" ordering, in which the vertices of the mesh are colored alternately red and black (with two adjacent nodes always having different colors), and then the matrix is permuted to put all the red nodes before all the black nodes. You should write a Matlab routine to produce this permutation. It will be easier if you always take k to be odd, which is all right for this homework.

For each of these orderings, you should explore the following preconditioning methods:

- No preconditioning.
- Precondition A with IC0 (via Matlab's `cholinc` routine).
- Precondition A with MIC (via Matlab's `cholinc` routine, with different parameters). Matlab's code doesn't produce an MIC0 ordering (I think), but it does produce a drop-tolerance version. You should experiment with a range of drop tolerances; part of the object of this part of the problem is to explore the drop-tolerance tradeoff between reducing the number of CG iterations (because the preconditioner is better) and increasing the number of operations per iteration (because the preconditioner is denser).
- Optional: Also experiment with preconditioning based on your ILU code from Homework 3.

Use conjugate gradient, via Matlab's `pcg` routine, to solve the system $Ax = b$ to a tolerance of 10^{-8}. Do this for as large a range of values of k as you can (all odd, if you wish). In each case, you should record both the number of CG iterations to convergence and an estimate of the number of flops per CG iteration (which depends on the number of nonzeros in the preconditioner). Use log-log plots (generated by Matlab) to estimate the number of iterations and the number of flops as functions of n.

Compare your results to some of the entries in the table of complexities from the April 30 class. Also, summarize your conclusions about the interacting effects of ordering permutations and drop-tolerance fill. Do different orderings perform better and different levels of drop tolerance?