CS 219: Sparse matrix algorithms: Homework 5

Assigned April 30, 2018

Due by class time Monday, May 7

Problem 1. You may use Matlab for this problem or do it by hand. I'm not sure which is easier.

Let A be the 9-by-9 matrix of the two-dimensional model problem on a 3-by-3 grid, with all diagonal elements equal to 4. This matrix is generated, for example, by `A = grid5(3)` in Matlab.

1(a) Show A factored as $A = UU^T$, where U is a matrix that has (i) 9 rows, (ii) no more than 2 nonzeros per column, and (iii) if a column has two nonzeros a and b, then $a = -b$. Note that U is not square.

1(b) Show the graph of the matrix A, with weights (all 1) on the edges for the nonzero values. Show a maximum-weight spanning tree for that graph. (There are a lot of choices since the weights are all the same.) Show the matrix B that is the support-graph preconditioner corresponding to the spanning tree you chose.

1(c) Show B factored as $B = VV^T$, where V is a matrix that satisfies the same three conditions as U above.

1(d) Find a matrix W with $U = VW$. Note that W is probably not square either.

1(e) What are the 2-norm, the 1-norm, the ∞-norm and the Frobenius norm of W? What is $\|W^TW\|_1$? What is the condition number of A? What is the condition number of $B^{-1}A$? (Answers computed by Matlab are ok.)

Problem 2. 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 `amd`.
- 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. Your Matlab routine can be almost a 1-liner 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, incomplete Cholesky with no fill, via Matlab's `ichol` routine.
- Precondition A with MIC, IC0 with modifications to the diagonal elements of the factor that preserve row sums in the preconditioner. (Here again you can use Matlab's `ichol` routine, with different parameters).
- Experiment with a range of drop tolerances. 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).

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 the number of CG iterations to convergence; the number of nonzeros in the incomplete factor; 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 how the number of iterations and the number of flops scale 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 at different levels of drop tolerance?

Problem 3 (extra credit). Experiment with BiCGSTAB and with GMRES on the nonsymmetric matrices in your menagerie from Homework 3. Try out different nonsymmetric preconditioners using Matlab's `ilu`. For GMRES, try out different restart parameters. In addition to plots and tables, write a couple of paragraphs to summarize what you discover. What advice would you give to a scientist who comes to you with a large nonsymmetric linear system to solve?