CS 219: Sparse matrix algorithms: Homework 3

Assigned April 16, 2018

Due by class time Monday, April 23

The Appendix contains definitions and pointers to references for terminology and notation.

Problem 1: Unsymmetric matrix menagerie. The purpose of this problem is for you to identify a set of a dozen or so unsymmetric matrices that span a range of characteristics, for use in testing and analyzing different approaches to solving unsymmetric linear systems.

Explore the SuiteSparse matrix collection and choose a set of 10 to 15 matrices. Try to choose a diverse set, from different collections and with different properties. Every matrix in your set should satisfy all the following criteria:

1. square (same number of rows as columns)
2. dimension between 500 and 50000
3. not symmetric
4. full structural rank (`sprank(A) == size(A,1)`)
5. Matlab can do `x = A\b` in less than a minute

Your set should contain a few matrices of each of the following types:

1. nearly strong Hall (few or one diagonal blocks in `dmperm`) and diagonally dominant
2. nearly strong Hall and not diagonally dominant
3. far from strong Hall (many diagonal blocks)

You can use the Matlab function `cs_dmspy()` to visualize the block triangular form of a matrix and see how nearly strong Hall it is. A good way to look for matrices that are far from strong Hall is to search the SuiteSparse collection for matrices with low "structural symmetry" (say less than 50%). However, be careful that you are only picking matrices with full structural rank; many of the highly nonsymmetric matrices in the collection have less than full rank.

Compute the following statistics for each matrix A in your set, and make a table of the results.

1. dimensions: `size(A)`
2. nonzeros: `nnz(A)`

3. structural symmetry:

```
B = A-diag(diag(A));
nnz(B & B')/nnz(B)
```

4. numeric symmetry:

```
Asymm = (A+A')/2;
Askew = (A-A')/2;
(norm(Asymm,'fro') - norm(Askew,'fro')) / norm(A,'fro')
```

5. number of blocks in Dulmage-Mendelsohn decomposition:

```
[p q r] = dmperm(A);
length(r) - 1
```

6. fraction of lower triangle within diagonal blocks:

```
[p q r] = dmperm(A);
sum(diff(r).^2) / prod(size(A))
```

7. column diagonal dominance:

```
B = abs(A);
d = diag(B);
min( d' ./ sum(B-diag(d)) )
```

8. row diagonal dominance:

```
B = abs(A');
d = diag(B);
min( d' ./ sum(B-diag(d)) )
```

Problem 2: Column intersection graph. Let A be n-by-n, square, and nonsingular. See the Appendix for the definition of the column intersection graph $G_\cap(A)$.

2a. Suppose A has a triangular factorization $A = LU$ without pivoting; that is, no row or column permutations are needed to avoid dividing by a zero diagonal element during Gaussian elimination. Prove that $G(L+U) \subseteq G_\cap^+(A)$; in other words, every directed fill edge that arises in $G(L)$ or $G(U)$ corresponds to an undirected fill edge that arises in the symmetric Cholesky factorization of the column intersection graph $G_\cap(A)$.

Hint: Play the directed graph game for vertices $j = 1 : n$ on the directed graph $G(A)$, and also play the Cholesky graph game for vertices $j = 1 : n$ on the undirected graph $G_\cap(A)$. Proceed by induction on j.

2b. Prove that, if P is any row permutation, $G_\cap(A) = G_\cap(PA)$; in other words, the column intersection graph is independent of the ordering of the rows of the matrix.

Theorem, not proved here: In the absence of numerical cancellation, if A is strong Hall, then for every edge (i, j) with $i > j$ of the filled column intersection graph $G^+_\cap(A)$, there exists a row permutation P such that $U(i, j)$ is nonzero in the factorization $PA = LU$. In other words, for strong Hall matrices the upper bound in Problem (2a) on the nonzero structure of U is the tightest possible given only the nonzero structure of A.

2c (extra credit). Exhibit a specific matrix A for which the upper bound $G(L+U) \subseteq G^+_\cap(A)$ is not tight for U; that is, $G^+_\cap(A)$ has an edge (i, j) with $i > j$, but for every row permuation of A that has a factorization $PA = LU$, element $U(i, j) = 0$.

Problem 3: Orderings for unsymmetric LU factorization. The moral of problem 2 is that a good *symmetric* permutation for the rows and columns of the (symmetric) matrix A^TA (i.e. for the column intersection graph $G_\cap(A)$) might be expected to be a good permutation for just the *columns* of A, if the rows are going to be permuted unpredictably by numerical partial pivoting. The idea of this problem is to see how well this works for the three different classes of matrices you identified in problem 1.

3a. For this part, use Matlab's `[L,U,P] = lu(A(:,q))`. This permutes the columns of matrix A according to permutation vector q, and then computes the LU factorization with partial pivoting using the left-looking triangular-solve algorithm presented in class on April 16.

Try factoring each of the dozen or so matrices in your test set with each of the following column permutations:

1. `q = amd(A'*A)`, which is a good symmetric ordering on $G_\cap(A)$.
2. `q = colamd(A)`, which is a similar ordering that Matlab can compute without forming the matrix A^TA.
3. `q = amd(A|A')`, which is a symmetric ordering that should be good if only there were no numerical pivoting in the factorization. In this case, you might try using `[L,U,P] = lu(A(q,q))` to preserve the diagonal of A.

For each experiment, report two complexity results:

1. Number of nonzeros in factors. This is just `nnz(L+U)`.
2. Number of flops in factorization. It's not hard to prove (try it!) that the number of nonzero floating-point operations needed to compute the factorization $PA = LU$ is the same as the

number of nonzero floating-point operations needed to multiply L by U (to get back PA). We know from Homework 2, problem 1, that this is equal to

$$\sum_i (\texttt{nnz}(\texttt{L}(:,\texttt{i})) * \texttt{nnz}(\texttt{U}(\texttt{i},:))).$$

Can you draw any conclusions about the suitability of different orderings for different classes of nonsymmetric problems?

3b. Repeat the experiment in (3a) above, but use the Matlab call `[L,U,P,Q] = lu(A(:,q))`. Instead of pure partial pivoting (where the column permutation is for sparsity and the row permutation is for numerics), this uses a more complicated strategy that tries to balance sparsity and numerics in both the row and column permutations. What conclusions can you draw?

Problem 4 (extra credit): Block triangular solve.

4a. (See Davis problem 7.4.) Write a Matlab function that solves $Ax = b$ by using the output of `dmperm(A)` to permute A to block upper triangular form (the same form displayed by `cs_dmspy(A)`), and then doing a block triangular backsolve to get x from b. The block backsolve should use Matlab's backslash operation once for each diagonal block of the permuted matrix. Confirm that your function is correct by comparing its results Matlab's.

Hint: In Matlab, if `I` and `J` are vectors of indices, then `A(I,J)` is a submatrix and/or permutation of `A` with just rows `I` and columns `J` in the specified order. With careful use of indexing notation, the block solve routine is only about a dozen lines of Matlab.

4b. Do an experiment similar to (3a) above, evaluating for each matrix the complexity in fill and flops of the factorization. This time, however, you should count fill and flops according to the block triangular form, factoring only the diagonal blocks. Thus, for each matrix and each column ordering method, you will first permute the entire matrix to block triangular form as in (4a); then factor each diagonal block $A_i = L_i U_i$ individually after permuting just the block A_i by the chosen column ordering method; then report total fill and total flops as follows.

1. fill: total number of nonzeros in each diagonal block's individual factors $L_i + U_i$, plus the total number of nonzeros in all the off-diagonal blocks of the original matrix A.
2. flops: total number of flops to compute all the individual $A_i = L_i U_i$ factorizations.

What conclusions can you draw for the various classes of matrices in your test set?

Appendix: Definitions and references.

- A matrix is *column diagonally dominant* if the absolute value of every diagonal element is larger than the sum of the absolute values of the other elements in its column.
- The *structural rank* of a matrix A is the maximum number of nonzeros that can be placed on the diagonal of A by row and column permutations PAQ, or equivalently the size of the largest matching in the bipartite graph of A. See Davis 7.2 and Matlab function `sprank()`. A square nonsingular n-by-n matrix must have structural rank n, also called *full structural rank*.

- The *column intersection graph* $G_\cap(A)$ of an m-by-n matrix A is an undirected graph with n vertices, one for each column of A, with an (undirected) edge (i, j) whenever columns i and j have nonzeros in some common row k. In the absence of numerical cancellation, $G_\cap(A)$ is the same as $G(A^T A)$, the graph of the symmetric n-by-n matrix $A^T A$. (This is because the (i, j) element of $A^T A$ is the dot product of columns i and j of A, which is zero if the nonzero structures of those columns don't intersect.) The definition and an example are in the posted class slides for April 16. The definition works whether or not A is square.

- The *filled column intersection graph* $G_\cap^+(A)$ is the result of playing the Cholesky graph game on $G_\cap(A)$; it's also the ordinary filled graph $G^+(A^T A)$. See April 16 class slides.

- For *block triangular form* see Davis 7.3 and Matlab function `dmperm()`.

- An m-by-n matrix is *strong Hall* (see Davis 7.3 and the April 16 class slides) if every set of k columns has nonzeros in at least $k + 1$ rows, for every k in the range 1 to $n - 1$. A square matrix is strong Hall if it can't be permuted to a nontrivial block triangular form. See Matlab function `cs_dmspy()`.

- The *Dulmage-Mendelsohn decomposition* of a matrix is a generalization of block triangular form. See Davis 7.3, Matlab functions `dmperm() and cs_dmspy()`, and the April 16 class slides. Computing the DM decomposition of a matrix amounts to first finding a maximum-size matching in a bipartite graph associated with the matrix, and then finding the strongly connected components of the directed graph of a permuted version of the matrix.