Math 128a - Program 3 - Due May16, 11am

In this programming assignment we will explore the accuracy with which three versions of Gaussian elimination can solve systems of linear equations:

1. Gaussian elimination with no pivoting (GENP),
2. Gaussian elimination with partial pivoting (GEPP) (the "standard" version, as used in Matlab)
3. Gaussian elimination with complete pivoting (GECP).

As we saw in class, GENP can fail on simple matrices. GECP is more expensive than GEPP, and usually no more accurate (which is why GEPP is the "standard"), but as we will see there are some hard examples where GECP does much better than GEPP.

The first part of your assignment will be to implement GECP in Matlab. In other words, you will write a function [x,PG] = gecpsolve(A,b) which returns the solution x of A*x=b computed using GECP, and another quantity defined below called the *pivot growth*. Second, you will test all 3 algorithms on a sequence of linear systems using some test code called tstGE.m that I supply; you only need to write gecpsolve. Then you will answer some questions about the results printed out by the testcode.

The test code is available in the usual place at `www.cs.berkeley.edu/~demmel/ma128a_Spr02/Homework/homework.html`. There are four routines there:

1. matgen.m - generates test matrices
2. genpsolve.m - solves A*x=b using GENP
3. geppsolve.m - solves A*x=b using GEPP
4. tstGE.m - runs tests on GENP, GEPP and GECP, and prints results.

The test code tstGE.m works as follows.

1. It generates 25 test matrices A, 5 different kinds (from easy (to solve accurately) to hard), each of 5 dimensions (5, 10, 25, 50, and 100). It also generates a random x and computes b = A*x, so we know the solution of A*x=b.
2. It solves A*x=b using GENP, GEPP and GECP (you need to supply the code for GECP). The results are called x_NP, x_PP and x_CP, respectively.
3. For each algorithm, it computes the "pivot growth" (PG_NP, PG_PP and PG_CP), which we define as the largest entry in L times the largest entry in U divided by the largest entry in A. This measures whether after Gaussian elimination LU is close to A (where we mean A after the rows and columns of A are possibly permuted by any pivoting.)
4. It computes the error Err_NP = $\| x - x_NP \| / \| x \|$, where $\| \cdot \|$ is the 2-norm. We expect Err_NP to be no smaller than about 10^{-16}, since that is machine epsilon, and all roundoff errors are this size. For very difficult matrices, Err_NP can be equal to 1 or larger, meaning that x_NP is completely inaccurate. Err_PP and Err_CP are defined analogously.

5. It computes an error bound Ebnd. We will (hopefully) discuss the theory behind this error bound in class, but let's just say here that it should bound the actual error Err provided that Gaussian elimination does not experience "pivot growth", i.e. PG is not $\gg 1$, which is what pivoting in GEPP and GECP attempts to avoid.

6. It computes an error ratio R_NP = Err_NP / Ebnd; R_PP and R_CP are defined analogously. If Ebnd really is an error bound, then R_NP should be at most about 1 (it may be 10 times or so greater due to accidents of roundoff). When it is much larger than 1, this means that there has been pivot growth. We also hope that R_NP is not much smaller than 1, since otherwise the true error is much smaller than the error bound, and the error bound is too pessimistic, perhaps due to accidents of roundoff.

7. Finally, it computes R_PG_NP = Err_NP / Ebnd / PG_NP; R_PG_PP and R_PG_CP are defined analogously. This error ratio takes pivot growth into account, and so should always be less one.

Here is your assignment in more detail.

1. Write gecpsolve.m in Matlab, which should solve A*x=b using GECP. Document it well according to class standards, and convince me that it is implemented correctly by running a few small test cases, and confirming that the L and U factors have the following properties:

   (a) The diagonal of L is all ones, and the subdiagonal entries of L are all bounded by 1 in absolute value.

   (b) The magnitude of the diagonal U(i,i) bounds the offidiagonal entries U(i,j) in the same row.

   In addition, the test cases in tstGE.m will also test your routine.

2. Run tstGE.m using your gecpsolve.m. Print out the results, and answer the following questions:

   (a) Which examples (numbered 1 through 25) had error bounds Ebnd bigger than .001, which means Ebnd predicts 3 or fewer correct digits of accuracy in the computed solutions? These matrices are called "ill-conditioned", and it is inherently difficult to solve with them accurately.

   (b) Were there any examples where Ebnd was much smaller than Err_NP, i.e. where GENP was much less accurate than predicted by Ebnd? Were any of these "easy" examples, in the sense that Ebnd was quite small, and either GEPP or GECP got them right? On which examples was there a lot of pivot growth for GENP? Is there a correspondence between these examples? How did Err_NP compare to Ebnd*PG_NP on these examples? Is GENP a generally reliable algorithm? Is Ebnd a generally reliable error bound for GENP (i.e. never much smaller than the true error, and rarely much larger? Is Ebnd*PG_NP a generally reliable error bound, in the same sense?

   (c) Were there any examples where Ebnd was much smaller than Err_PP, i.e. where GEPP was much less accurate than predicted by Ebnd? Were any of these "easy" examples, in the sense that Ebnd was quite small, and GECP got them right? On which examples was there a lot of pivot growth for GEPP? Is there a correspondence between these examples? How did Err_PP compare to Ebnd*PG_PP on these examples? Is GEPP a generally reliable algorithm? (Note: case=4 is essentially the only known example where

GEPP has large pivot growth). Is Ebnd a generally reliable error bound for GEPP (i.e. never much smaller than the true error, and rarely much larger? Is Ebnd*PG_NP a generally reliable error bound, in the same sense?

(d) Were there any examples where Ebnd was much smaller than Err_CP, i.e. where GECP was much less accurate than predicted by Ebnd? On which examples was there a lot of pivot growth for GECP? Is GECP a generally reliable algorithm? Is Ebnd a generally reliable error bound for GECP (i.e. never much smaller than the true error, and rarely much larger? Is Ebnd*PG_CP a generally reliable error bound, in the same sense?