Lab Session 2

MA- 423 : Matrix Computations September - November, 2020 S. Bora

---

1. Use the MATLAB function program `[L,U] = genp(A)` to do the following:

   (a) Find the factors L and U of an LU decomposition of $A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}$. What is $A-LU$?

   (b) Solve the system of equations $Ax = b$ where $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ by using the computed LU factorization from `genp` and the programs `rowforward` and `colbackward` in Lab 1 for forward and backward substitution in the correct order. What is the difference of your answer with the correct solution in the 2-norm?

   What can you conclude about GENP from the above algorithm? Can you identify the step at which things start to go wrong?

2. Write a function program `[L,U,p] = gepp(A)` to find a unit lower triangular matrix `L`, an upper triangular matrix `U` and a column vector `p` satisfying `A(p,:) = LU` via Gaussian Elimination with Partial Pivoting (GEPP) When doing so please note the following:

   (a) Your code should make only the most minimal changes to the `genp` code written in Lab 1. In particular it should retain all important features of `genp` that ensure efficiency.

   (b) The built in Matlab function program `lu` performs GEPP and GECP to find LU decompositions of appropriately permuted matrices, also giving the permutation matrices used in each case. Type `help lu` for details. Compare the output of your `gepp` code with the corresponding outputs of the `lu` program. *The comparision should be performed for several different randomly generated matrices (use* `randn` *command for this).*

3. Write a function program `x = geppsolve(A,b)` to solve a system $Ax = b$ via GEPP. Your program should call the program `[L,U,p] = gepp(A)` and the programs written in Lab 1 for solving upper and lower triangular systems. Compare your answers with that of the MATLAB command $A\backslash b$ (which uses GEPP to solve the system) for several different choices of A and b that are randomly generated by using the `randn` command.

4. Given $A \in \mathbb{R}^{n\times n}$, write a function program `d = mydet(A)` that uses an *efficient* version of LU factorization to compute the determinant of A in $O(n^3)$ flops.

5. Write a function program `G` $=$ `mycholb(A)` that executes the *bordered form* of the Cholesky Decomposition for finding the Cholesky factor of an $n \times n$ positive definite matrix A in $\frac{n^3}{3} + O(n^2)$.

   Compare your output with that of the built in Matlab function program `chol` for several different choices of randomly generated positive definite matrices. The following commands may be used to generate them with arbitrary choices of $0 < a < b$, and positive integers n :

   $\gg$ `r = a + (b-a).*rand(n,1); D = diag(r)`
   (Here `r` is a length `n` column vector of values randomly generated from an uniform distribution on the interval `[a,b]` and `D` is an $n \times n$ diagonal matrix with the entries of `r` on the diagonal.)
   $\gg$ `B = randn(n); [Q,R] = qr(B)`
   (Here B is an $n \times n$ matrix containing pseudorandom values drawn from the standard normal distribution and Q is an orthogonal matrix such that $B = QR$ is a QR decomposition of B.)
   $\gg$ `A = Q'*A*Q.`

(The positive definite matrices are also precisely symmetric matrices with positive diagonal entries and the above commands generate them randomly. Wait for some more classes to know why!)