Netlib ← link for programs
.org

EE 615

Project 1

Due: March 2, 2016

1. Task: To develop 2 computer codes implementing routines for LU factorization and conjugate gradient algorithms. You can either develop completely new routines or modify existing routines in the public domain (e.g., LINPACK, LAPACK and SLAP in NETLIB (http://www.netlib.org), Numerical Recipes by W.H. Press, et al., in-house software, etc.) In any case, I would like to encourage you to look at the routines in NETLIB, since some of their routines are optimized in certain sense. Also, you may need those routines for future projects, so you may like to keep that in mind. Double precision real or single precision complex arithmetic will be good enough.

2. General Requirements

A) The LU factorization code should give L & U with either diagonal of 1's in [U] or diagonal of 1's in [L]. For the first case, the equations are given in the notes. For the latter case, the equations are

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad (u_{11} = a_{11})$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}}$$

Forward Substitution: make $l_{mm} = 1$ on p. 8 of notes

Backward Substitution: equations on p. 9 of notes become

also the code should includ pivting

$$x_N = \frac{c_N}{u_{NN}}$$

$$x_m = \frac{c_m - \sum_{i=m+1}^{N} u_{mi} x_i}{u_{ii}}, \quad m = N-1, N-2, \ldots 1$$

The code should provide pivoting, forward substitution, and backward substitution. Also, see Appendix A on pivoting.

B) The CG code should be able to handle a Jacobi preconditioner and nonsymmetric / non-positive-definite system.

3) You should hand in

A) Hard copy of your programs documenting the major steps in your algorithm. If you are modifying an existing code, you should give the source of your original code and describe the major things, e.g., error checking, that are not mentioned in class.

B) Source Codes

C) Results

i) Files for dense matrices will be provided by email or other means.

1) LU_test.in — 5 × 5 matrix with some zeros on the diagonal for testing LU routine

2) CG_test.in — 5 × 5 nonsymmetric matrix for testing CG routine

3) LU.in - 128 x 128 matrix for LU routine
4) CG.in - " " for CG "

All the above input files have the following format:

a) One number per line
b) first number is the dimension of the matrix
c) the matrix is given row by row

For example, for the matrix

$$\begin{bmatrix} 3.4 & 9.0 & 2.1 \\ 4.5 & 2.1 & 7.8 \\ 9.4 & 6.5 & 3.2 \end{bmatrix}$$

the file will look like

3x3 ← (3)
3.4 ✓ i
9.0
2.1
4.5
2.1
7.8
9.4
6.5
3.2

If that says 128 so its 128x128

ii) Hand in the following: for $\bar{b} = [1, 1, 1, 1, \cdots 1]^T$

a) For LU Routine — [L] & [U] & $\bar{x}$ for LU.in

b) For CG Routine — $\|\bar{r}\|^2 = \|\bar{b} - [A]\bar{x}\|^2$ for the final $\bar{x}$, number of iterations, $\bar{x}$ for CG.in

(You should try to see whether you can get $\|\bar{r}\|^2$ down to 10^{-3} or less)

c) An estimate of the dynamic memory & solution time for the LU and CG codes. You can use the "top" command to get the dynamic memory and the "time" command to get the CPU time. (Some other means

such as "dtime" can also be used.)