ME 504 Numerical Methods for Engineering Computation

Assignment 3

(System of Linear Equations)

Max Marks : 45

Note :

The assignments will use different test matrices. The data is large and hence it is shared through iitmandi cloud https://cloud.iitmandi.ac.in/d/60a8c9822b/. (if you have any issues accessing the data, can contact TA's.)

Following matrices are provided.

1. tri_diag10, tri_diag1000, tri_diag10000 – tridiagonal matrices *'a'* of different size in dense/full format.

2. k_1941, k_11943 – general matrices and their corresponding load vector/'b' vector f_1941 and f_11943

Following commands can be used to import data (.txt files) to Matlab/Octave workspace – importdata, load, dlmread, csvread, textscan etc.

For seeing the data in text files, you can do it using workpad/notepad, however, for large matrices, your system will hang. A software that may provide an edge over this is 'notepad++'.

In working with large systems, it is recommended to first test your code with small problem (say 10 x 10 or 5x5) that can be debugged easily. Then the code can be executed on bigger set of data.

---

1. (a) Load the test matrices provided in cloud folder (one at a time). For visualizing matrices, use command *spy(matrix)*. In your assignments, you must have spy plots for the given matrices.
(for large some, the system may hang! Be honest in your reporting.)
(b) For data set 1 (tri-diagonal matrices), calculate the computational time for solving '$ax = b$' using following solvers - Gauss Elimination, TDMA, matlab '\' solver, Jacobi, Gauss-Seidel, SOR (omega as 1.2). For each test data, compare the computational time using these six methods. **Plot the result of computational time as bar chart for better visualization of results.** The load vector b can be taken as ones(n,1), where n x n is size of the matrix. Also plot relative error norm(Current iteration value - Previous iteration value) with iteration number for iterative solvers.
(c) For data set 2 (general matrices k_...., f_....), you may use same methods except TDMA being replaced by Conjugate Gradient method. The corresponding load vector for these cases are provided in shared folder. Again compare the computational time by six methods [3+8+9]

2. Hilbert matrix can be used to test algorithm. The Hilbert matrix coefficients are given as $a_{ij}=(i+j-1)^{-1}$. For a known solution of x = [1, 1, ... 1]^T, the b vector can be calculated as $b_i=\sum_{j=1}^{n} a_{ij}$. Hilbert matrix can be generated in matlab/octave using hilb command.

Solve the above system of equations $\sum_{j=1}^{n} a_{ij} x_j = b_i$ for x using Gauss Elimination.

Find residue (as norm of (b – ax)) and absolute error (as norm(x_exact – x_calculated)). Tabulate the data of residue and absolute error for $3 \le n \le 15$. Comment upon increase in (abs)error for large values of n. [5]

4. Consider a sample two-dimensional linear system $Ax = b$ as

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 2 \\ -8 \end{Bmatrix}$$

A, b are mentioned above, $c = 0$.
Plot graphs to show the following.

a. Solution lies at intersection of two lines.
b. Graph the quadratic form

$$\boldsymbol{F}(\boldsymbol{x}) = c + b^T x + \frac{1}{2} \boldsymbol{x}^{\boldsymbol{T}} \boldsymbol{A} \boldsymbol{x}$$

showing that the minimum point of this surface is the solution of Ax = b

c. Contours of the quadratic form so that each ellipsoidal curve has a constant value

d. Gradient $F'(x)$ of the quadratic form. Show that for every x, the gradient points in the direction of the steepest increase in $F(x)$ and is orthogonal to the contour lines. [10]

5 Using Jacobi, Gauss-Seidel and the SOR Method ($\omega = 1.4$) iterative method, write and run code to solve the following linear system to four decimal places of accuracy.

$$\begin{bmatrix} 7 & 3 & -1 & 2 \\ 3 & 8 & 1 & -4 \\ -1 & 1 & 4 & -1 \\ 2 & -4 & -1 & -6 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{bmatrix} -1 \\ 0 \\ -3 \\ 1 \end{bmatrix}$$

Compare the number of iterations in each case. [5]

6. Solve the above problem using the SOR iterative method with values of $\omega = 1:0.1:2$. Plot the number of iterations for convergence verses the values of ω . Which value of ω results in fastest convergence. [5]