DEPARTMENT OF PHYSICS

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS

PH5720 Numerical Methods and Programming Lab-Week-05 21 Feb 2020

Time: 2:00 pm - 5:00 pm

Goal of this session:

1. Write your own Gaussian Elimination Code (with and without pivoting).
2. Write your LU decomposition code to solve the set of linear equations.
3. Solution of a tridigoanal matrix linear equations.
4. Advantage of using dynamical memory allocation (pointer) compared to static memory allocation (array).

I. NUMERICAL PROBLEMS

1. Consider the linear system of equations

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = w_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = w_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = w_3.$$

This can be written in matrix form as

$$\mathbf{A}\mathbf{x} = \mathbf{w}.$$

We specialize here to the following case

$$-x_1 + x_2 - 4x_3 = 0$$

$$2x_1 + 2x_2 = 1$$

$$3x_1 + 3x_2 + 2x_3 = \frac{1}{2}.$$

Obtain the solution (by hand) of this system of equations by doing Gaussian elimination.

2. Write therafter a program which implements Gaussian elimination (with pivoting) and solve the above system of linear equations. How many floating point operations are involved in the solution via Gaussian elimination without pivoting? Can you estimate the number of floating point operations with pivoting?

3. We are going to solve the onedimensional Poisson equation with Dirichlet boundary conditions by rewriting it as a set of linear equations. The 1D Poisson equation takes the form:

$$\frac{d^2\phi}{dx^2} = -\frac{\rho(x)}{\epsilon_0} \tag{1}$$

where $\phi(x)$ satisfies the boundary conditions $\phi(0) = 0$ and $\phi(d) = V_0$. Consider for example a vacuum diode, in which electrons are emitted from a hot cathode and accelerated towards an anode. The anode is held at a large positive potential V_0 with respect to the cathode. We can think of this as an essentially onedimensional problem. Suppose that the cathode is at $x = 0$ and the anode at $x = d$.

In our case we will rewrite Poisson's equation in terms of dimensionless variables. We can then rewrite the equation as:

$$\text{-}u''(x) = f(x), \quad x \in [0,1], \quad u(0) = u(1) = 0.$$

and we define the discretized approximation to u as v_i with grid points $x_i = ih$ in the interval from $x_0 = 0$ to $x_{n+1} = 1$. The step length or spacing is defined as $h = 1/(n+1)$. We have then the boundary conditions $v_0 = v_{n+1} = 0$. We approximate the second derivative of u with

$$-\frac{v_{i+1}-2v_i+v_{i-1}}{h^2} = f_i, \quad \text{for } i = 1, 2, ..., n,$$

where $f_i = f(x_i)$. Show that you can rewrite this equation as a linear set of equations of the form $\mathbf{Av} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ tridiagonal matrix which we rewrite as

$$A = \begin{pmatrix} 2 & -1 & 0 & ... & ... & 0 \\ -1 & 2 & -1 & 0 & ... & ... \\ 0 & -1 & 2 & -1 & 0 & ... \\ 0 & 0 & -1 & 2 & -1 & ... \\ & ... & ... & ... & ... & \\ 0 & ... & & -1 & 2 & -1 \\ 0 & ... & & 0 & -1 & 2 \end{pmatrix}$$

and $b_i = h^2 f_i$.

In our case we will assume that $f(x) = (3x + x^2)e^x$, and keep the same interval and boundary conditions. Then the above differential equation has an analytic solution given by $u(x) = x(1-x)e^x$. We will compare our numerical solution with this analytic result in the next problem.

4. We can rewrite our matrix **A** in terms of onedimensional vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ of length $1 : n$. A tridiagonal matrix is a special form of matrix where all the elements are zero except for those on and immediately above and below the leading diagonal. The above tridiagonal system can be written as

$$a_i v_{i-1} + b_i v_i + c_i v_{i+1} = b_i,$$

for $i = 1, 2, ..., n$. The algorithm for solving this set of equations is rather simple and requires two steps only, a decomposition and forward substitution and finally a backward substitution.

Your first task is to set up the algorithm for solving this set of linear equations. Find also the number of operations needed to solve the above equations. Show that they behave like $O(n)$ with n the dimensionality of the problem. Compare this with standard Gaussian elimination.

Then you should code the above algorithm and solve the problem for matrices of the size 10×10, 100×100 and 1000×1000. That means that you choose $n = 10$, $n = 100$, and $n = 1000$ grid points.

Compare your results (make plots) with the analytic results for the different number of

grid points in the interval $x \in [0,1]$. The different number of grid points corresponds to different step lengths h.

Compute also the maximal relative error in the data set $i = 1, ..., n$,by setting up

$$\epsilon_i = log_{10}\left(\frac{v_i - u_i}{u_i}\right),$$

as function of $log_{10}(h)$ for the function values u_i and v_i. For each step length extract the max value of the relative error. try to increase n to $n = 1000$ and $n = 10^5$. Comments your results.

5. Compare your results with those from the LU decomposition codes for the matrix of size 1000×1000. Use for example the unix function time when you run your codes and compare the time usage between LU decomposition and your tridiagonal solver. Can you run the standard LU decomposition for a matrix of the size $10^5 \times 10^5$? Comment your results.

6. Obtain $u(x)$ numerically using the gsl library functions gsl_linalg_LU_decomp and gsl_linalg_LU_solve. Compare the relative error between the analytical and the numerical solutions as a function of x for different N values by plotting the relative error as for $N = 10, 100, 1000$ as a function of x.

II. USE: FUNCTION CLOCK()

Using the C clock function clock() determine the time taken for solving the linear equation as a function of N. You will need to include the header file time.h. The clock function can be used as follows:

```
//Declaring variables start and stop of type clock_t,
which is defined in time.h\\

clock_t start, stop; \\

start = clock(); \\
```

```
        // Body of the code: \\

 stop = clock();\\

printf(XXXX )\\
```