SOLVING ONE DIMENSIONAL POISSON EQUATION

FYS 4150 – COMPUTATIONAL PHYSICS

JEYALAKSHMI THOPPE SUBRAMANIAN

https://github.com/jeyalakt/4150COMP_PHY.git

Abstract :

In this Project one dimensional Poisson equation with Drichlet boundary conditions is solved with two algorithms -Gaussian Elimination and LU decomposition . It is observed that Gaussian elimination algorithm consumes less time and hence efficient than the LU decomposition. LU decomposition algorithm was used via pre built library function could not be used for discretion values above in the order of 10000 while Gaussian elimination algorithm can be used .

Contents

1. Introduction

Poisson equation describes the electrostatic potential generated by localised charge distribution in electromagnetism . This study intents to solve the Poisson equation using two different algorithms -first, Gaussian elimination algorithm and second LU decomposition method.

We study how these two algorithms are used to solve the Poisson equation at different precision levels and how Gaussian elimination is more efficient that LU decomposition method.

First, the how the second derivative of Poisson equation can be approximated to 3 point formula is presented followed by elaboration on the proposed algorithms .Then the results of the algorithms are presented . Finally, a brief discussion on the results and remarks are provided.

2. Theory

2.1. The Poisson equation: In electromagnetism, the Poisson equation describes the electrostatic potential ϕ generated by localised charge distribution $\rho(r)$, by

$$\nabla^2 \phi = -4\pi \rho(r). \qquad - \text{①}$$

in three dimensions.

In this study we assume spherical symmetry of ϕ and $\rho(r)$. So the equation is simplified as one dimension equation dependant on radius r, as

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2 \frac{d\phi}{dr}\right) = -4\pi \rho(r) \qquad - \text{②}$$

substituting $\phi(r)$ with $\frac{\phi(r)}{r}$

$$\frac{d^2\phi}{dr^2} = -4\pi r \rho(r) \qquad - \text{③}$$

letting $\phi \to u$ and $r \to x$, this equation simplifies as

$$-u''(x) = f(x) \qquad - \text{④}$$

The inhomogeneous term f ~~is~~ or source term is given by the charge distribution ρ multiplied by r and constand -4π.

However, in this study the source form is taken as $f(x) = 100e^{-10x}$ — ⑤

and the results will be compared with numerical solution of

$$u(x) = 1-(1-e^{-10})x - e^{-10x} \quad — ⑥$$

In this project, we try to solve the one dimensional Poisson equation with Dirichlet boundary condition

$$-u''(x) = f(x) \quad , x \in (0,1) \quad u(0)=u(1)=0 — ⑦$$

2.2. Approximation of second derivative: In this study the one-dimensional Poisson equation is solved with Dirichlet boundary conditions by rewriting it as set of linear equations.

The discretised approximation of u is defined as v_i with discretion or grid points $x_i = ih$; step size $h = \frac{1}{n}$. in the interval $x_0 = 0$ to $x_n = 1$ with boundary conditions $v_0 = 0$ & $v_n = 0$.

n is normally taken in multiples of 10.

As the boundary conditions are fixed they are ignored and only the interior solution $v_i \in (1 \dots n-1)$ needs to be found.

The second derivative of equation (4) is approximated as 3 point formula

$$\frac{-v_{i+1} - v_{i-1} - 2v_i}{h^2} = f_i, \text{ for } i = 1 \ldots n \qquad - (8)$$

where $f_i = f(x_i)$

Now f is defined as ~~ff~~ $f = h^2 f_i$,
the equation can be rewritten as

$$-v_{i+1} + 2v_i - v_{i-1} = h^2 f_i \qquad - (9)$$

which can be represented in matrix equation

$$Av = f \qquad - (10)$$

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 \\ & \vdots & & \ddots & & & \\ 0 & 0 & 0 & 0 & & 2 & -1 \\ 0 & 0 & 0 & 0 & & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_n \end{bmatrix} \qquad - (11)$$

3. Algorithms

Two algorithms are used to solve eqn (10) and the results are compared.

Gaussian elimination of Tridigonal matrix A, is used first. This method is also called the Thomas Algorithm [1].

The second method is LU decomposition.

Complete programs in C++ and plotting of results in python program is given in the github link

3.1. Gaussian elimination Algorithm.

Notes from github repository of Computational physics for linear algorithms [2] are used to solve.

Matrix A in (11) is written is general form as:

$$\begin{bmatrix} d_1 & c_1 & 0 & \cdots & & 0 & 0 \\ a_2 & d_2 & c_2 & \cdots & & 0 & 0 \\ 0 & a_3 & d_3 & \cdots & & 0 & 0 \\ \vdots & & & \ddots & & d_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & & a_{n-1} & d_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \vdots \\ \tilde{b}_n \end{bmatrix} \quad - (12)$$

This is solved in two steps:
Forward substitution as follows:

$$\tilde{d}_i = d_i - \frac{a_i c_{i-1}}{\tilde{d}_{i-1}} \quad - \text{(13)}$$

$$\tilde{d}_1 = d_1$$

and

$$\tilde{b}_i = b_i - \frac{a_i \tilde{b}_{i-1}}{\tilde{b}_{i-1}} \quad - \text{(14)}$$

$$\tilde{b}_1 = b_1 = b_1$$

The final solution is given by step two.
Backward substitution.

$$v_{n-1} = \frac{b_{n-1}}{d_{n-1}} \quad - \text{(15)}$$

$$v_i = \frac{\tilde{b}_i - c_i v_{i+1}}{\tilde{d}_i} \quad - \text{(16)}$$

where i is from 1 to n-2

The time taken for this algorithm is calculated for different step size of n as the algorithm is run for various values of n.

3.2. LU decomposition:

This method involves writing matrix

$A = LU$ — (17)

where L – Lower Triangular matrix
U – upper Triangular matrix ~~—to~~

$A = LUv = f$ is solved in two steps.

$Lw = f$ — (18)

$Uv = w$ — (19)

Prewritten library function in Armadillo is used which solves linear equations as

solve $(Ax = b)$ — (20)

The time taken for various values of n is calculated.

It is noted that the program does not run for 10^5 and multiples further as it uses two much memory.

Also for both algorithms at different size of n, relative error between exact solution and numerical solution is calculated and maximum value of relative error as a function of step size is plotted (fig 5)

4. Snippets of algorithms

```
// Forward substitution
 clock_t start, finish;
       start = clock();
 for (int i = 2; i < n; i++)
      b(i)=b(i)-(a(i)*b(i-1)/d(i-1));
     //b(i) = b(i) + b(i-1)/d(i-1);
 // Backward substitution
 solution(n-1) = b(n-1)/d(n-1);
 for (int i = n-2; i > 0; i--)
     solution(i)=(b(i)-(c(i)*solution(i+1)))/d(i); // gen soln
     //solution(i) = (b(i)+solution(i+1))/d(i);
 finish = clock();
```

Snippet of Gaussian elimination algorithm

```
   clock_t start, finish;
   start=clock();
  for (int i = 1; i < n-1; i++){
    x(i) = x(i-1)+h;
b(i) = hh*f(x(i));
    A(i,i-1)  = -1.0;
    A(i,i)    = 2.0;
    A(i,i+1)  = -1.0;
  }
  A(n-1,n-1) = 2.0; A(n-2,n-1) = -1.0; A(n-1,n-2) = -1.0;
// solve Ax = b
  vec solution  = solve(A,b);
  finish = clock();
```

Snippet of LU decomposition with armadillo built-in function

5. Results

The results in terms of step size , exact function , numerical solution , relative error are stored in a file for each value of n .

CPU time comparision

The timings of the algorithms were observed to be of the following order

(Note: timing includes memory read and write and on the other tasks being processed by CPU at that time)

No. of points	CPU time(sec)	
	Gaussian elimination algorithm	LU decomposition algorithm
10	0	0
100	0	0.001
1000	0	0.009
10000	0.002	0.764
100000	0.004	Prg fail

Table 1: comparison of CPU time for Gaussian elimination and LU decomposition algorithm

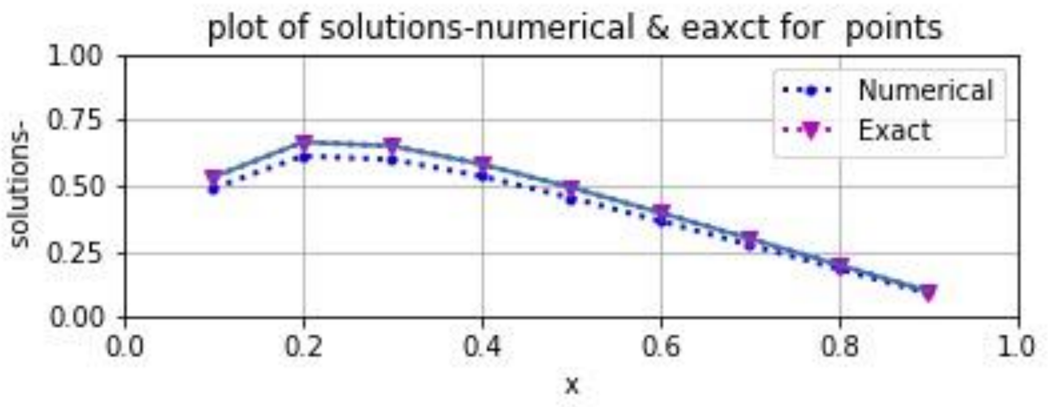

Figure 1 : Plot of exact and numerical solution by Gaussian Elimination algorithm for n=10 points

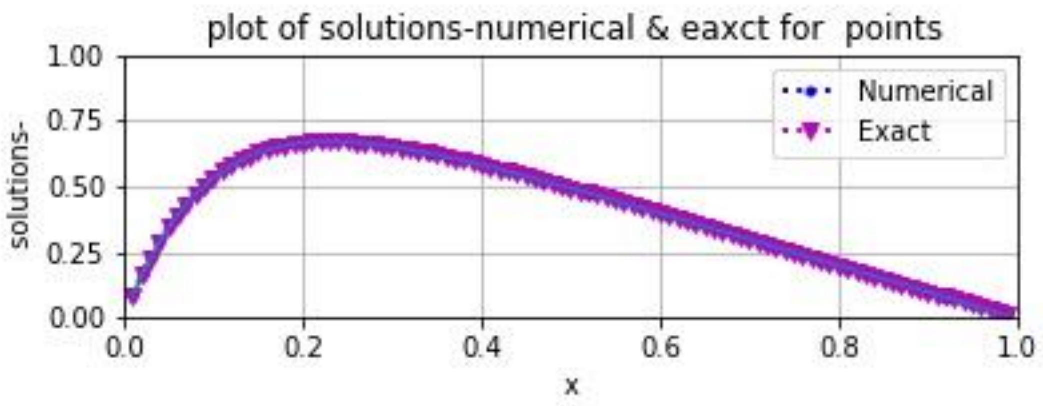

Figure 2 : Plot of exact and numerical solution by Gaussian Elimination algorithm for n=100 points

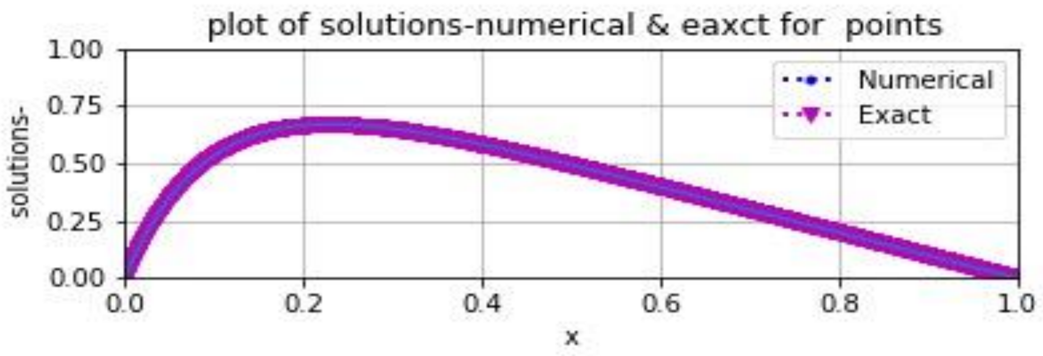

Figure 3 : Plot of exact and numerical solution by Gaussian Elimination algorithm for n=1000 points

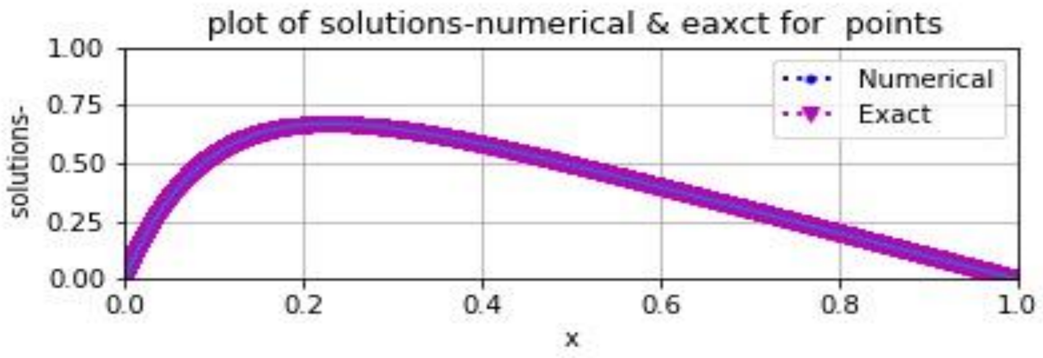

Figure 4 : Plot of exact and numerical solution by Gaussian Elimination algorithm for n=10000 points

Similarly, plots can be plotted for LU decomposition algorithm also.

No. of points(n)	CPU time(sec)	
	Gaussian elimination algorithm	LU decomposition algorithm
0	1.10058	1.1005822
100	3.0793984	3.0793984
1000	5.0791834	5.0791834
10000	7.0791987	7.0787883
100000	8.842964	Prg fail

Table 2: comparison of absolute value of relative error with Gaussian elimination and LU decomposition algorithm

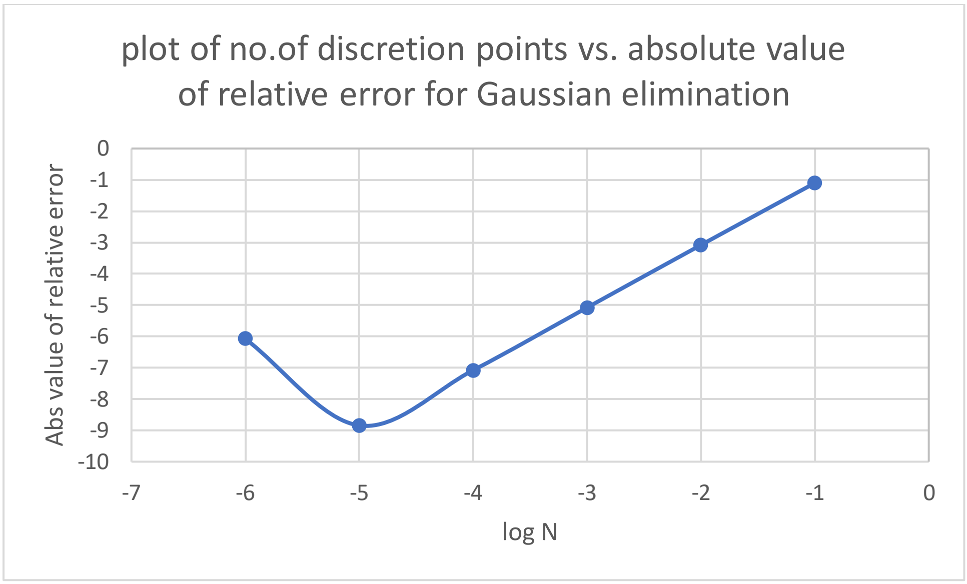

Figure 5 : plot of no.of discretion points vs. absolute value of relative error for Gaussian elimination

6. Discussion

Normally we intend to choose large value for n , assuming that it will result in higher precision .But from figure 5 (for gaussian elimination algorithm)we observe that this is true upto certain value of n only . This is due to loss of precision in numerical representation in the computer.A suitable value of n =10^5 gives results at reasonable cost of CPU time and memory

Also the LU decomposition algorithm consumes 8*10^{10} bytes (app. 80 GB) of memory for value n=10^5 which strongly affects our ability performing the calculation.

So Gaussian elimination method is found more efficient.

7.Conclusion

The results show the importance of choosing suitable algorithm before solving a problem numerically within the constraints of time and memory and at reasonably accepted precision. Loss of numerical precision in numerical representation in computer also needs to to be considered.

In this study it is found that Gaussian elimination algorithm is more efficient than LU decomposition .

References

[1] Thomas, L.H. (1949), Elliptic Problems in Linear Differential Equations over a Network. Watson Sci. Comput. Lab Report, Columbia University, New York.
[2] Hjorth-Jensen, M. (2015). Computational Physics - Lecture Notes 2015. University of Oslo
[3]] Hjorth-Jensen, M. (2015). Computational Physics-documents- linear algorithms. . University of Oslo