Project 1 - FYS4150

Sajjad Ahmadigoltapeh, Einar Aurbakken, Bastian Skjelstad, Per Harald Barkost
email-address: sajjadah@uio.no, eanorway@gmail.com, bastiabs@uio.no, ph.barkost@gmail.com
(Dated: September 10, 2018)

`https://github.com/einaur/fys4150/tree/master/Project1/main_code/C`
`https://github.com/einaur/fys4150/tree/master/Project1/main_code/Python`

We solve the one-dimensional Poisson equation with Dirichlet boundary conditions using Gaussian elimination. We use a solver for a general tridiagonal matrix, and then develop a specialized solver for the case of a symmetric Toeplitz matrix. Computations with different step-size shows that the relative error decreases with smaller step-size only until a certain limit. Comparison with a LU decomposition solver shows that the Gaussian solver specialized for tridigaonal matrices are significantly faster.

I. INTRODUCTION

Linear differential equation with the form of $u''(x) = f(x)$ is an ordinary equation in the scientific problems and Poisson's equation which is a wellknown second order differential equation in electromagnetism follows above mentioned form. In this study two solutions will be introduced for Poisson's equation which via using C++ programming language. First, one dimensional Poisson equation with Dirichlet boundary conditions was analyzed and solved with tridiagonal matrix algorithm. Afterward, LU decomposition method was applied to solve the same differential equation. Both methods are tested for different sizes of matrix and their relevant relative errors were calculated.

II. METHODOLOGY

A. Stating the problem

Poisson's equation is defined as equation(1).

$$\nabla^2 \Phi = -4\pi\rho(\mathbf{r}) \tag{1}$$

Using spherical dimension, the ∇^2 operator is defined as equation(2)

$$\nabla^2 = \frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\Phi}{dr}\right) \tag{2}$$

Using a spherically symmetric assumption simplifies above equation to equation(3) which depends only on radius r :

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

which is generally is presented as:

$$-u''(x) = f(x), \qquad x \in (0,1), \qquad u(0) = u(1) = 0 \tag{4}$$

Above mentioned boundary condition is called Dirichlet boundary condition. To employ Gaussian elimination method, first step is discretizing the domain of x. In fact, using Taylor expansion helps differential equation(4) is approximated by a linear equation with an error of $(\mathcal{O}(n^2))$. Namely, second order differential equation(4) now is presented as equation(5)

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} = f_i \qquad \text{for } i = 1, 2, ..., n \tag{5}$$

such that, mesh size is given as: $h = 1/(n+1)$ and $x = ih$ and relevant boundary conditions are $u_0 = u_{n+1} = 0$, where $f_i = f(x_i)$. Generally, a set of generated linear equations mathematically are shown as:

$$\boldsymbol{Au} = \tilde{\boldsymbol{b}} \tag{6}$$

where A is a tridiagonal matrix i.e.

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & \cdots \\ 0 & -1 & 2 & -1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & -1 & 2 \end{pmatrix} \tag{7}$$

and $\tilde{b_i} = -h^2 f_i$. Then, floating points and relevant timing of algorithm is tested for $n = 10, 100, 1000$ grid points.
To estimate the relative error of applied method, $f(x)$ should be calculated via numerical and exact solutions. Hence, first of all set of linear equations which is derived as equation (5) is solved. Then by supposing $f(x) = 100e^{-10x}$ for equation (4) that, have exact solution of: $u(x) = 1 - (1 - e^{-10})x - e^{-10x}$ it will be possible to estimate relative error.

B. General algorithm

General tridiagonal matrix is shown as equation (8):

$$\boldsymbol{Au} = \begin{pmatrix} a_1 & b_1 & 0 & \cdots & \cdots & 0 \\ c_2 & a_2 & b_2 & 0 & \cdots & \cdots \\ 0 & c_3 & a_3 & b_3 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & c_{n-1} & a_{n-1} & b_{n-1} \\ 0 & \cdots & \cdots & \cdots & c_n & a_n \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ \\ \vdots \\ d_n \end{pmatrix} \tag{8}$$

Using Gaussian elimination provides forward substitution as:

$$\boldsymbol{Au} = \begin{pmatrix} \tilde{a_1} & b_1 & 0 & \cdots & \cdots & 0 \\ 0 & \tilde{a_2} & b_2 & 0 & \cdots & \cdots \\ 0 & 0 & \tilde{a_3} & b_3 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & \tilde{a_{n-1}} & b_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & a_n \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} \tilde{d_1} \\ \tilde{d_2} \\ \vdots \\ \\ \vdots \\ \tilde{d_n} \end{pmatrix} \tag{9}$$

where $\tilde{a_1} = a_1$ and

$$\tilde{a_i} = a_i - b_i c_i / \tilde{a_{i-1}}, \qquad i = 2, 3, ..., n \tag{10}$$

and for right hand side:

$$\tilde{d_i} = d_i - c_i \tilde{d_{i-1}} / \tilde{a_i}, \qquad i = 2, 3, ..., n \tag{11}$$

The backward substitution i.e. $u_n = \tilde{d_n}/\tilde{a_n}$ gives:

$$u_i = (\tilde{d_i} - b_i u_{i+1})/\tilde{a_i}, \qquad i = n-1, n-2, ..., 1 \tag{12}$$

An important point for each algorithm which should be considered is how efficient the algorithm is. Therefore, it is worth to find how many floating point is required for an algorithm. In fact, for forward substitution 6 FLOPS for each iteration is counted which becomes $6 * (n-1)$ FLOPS for whole iteration. Furthermore, $\mathcal{O}(8n)$ FLOPS is required for backward substitution.

C. LU decomposition method

LU decomposition technique let write a non-singular matrix A as a product of L and U

$$A = LU \tag{13}$$

where L is a lower triangular matrix

$$L = \begin{pmatrix} l_{11} & 0 & 0 & \cdots & \cdots & 0 \\ l_{21} & l_{22} & 0 & 0 & \cdots & \cdots \\ l_{31} & l_{32} & l_{33} & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & l_{n-1,n-1} & 0 \\ 0 & \cdots & \cdots & \cdots & l_{n,n-1} & l_{nn} \end{pmatrix} \tag{14}$$

and U is a upper triangular matrix

$$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \ldots & u_{1n-1} & u_{1n} \\ 0 & u_{22} & u_{23} & \ldots & u_{2n-1} & u_{2n} \\ 0 & 0 & u_{33} & \ldots & u_{3n-1} & u_{3n} \\ & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & \ldots & u_{n-1n-1} & u_{n-1n} \\ 0 & 0 & 0 & \ldots & 0 & u_{nn} \end{pmatrix} \tag{15}$$

and finally A could be substituted with L and U as following:

(**note: to prevent any confusion for LU decomposition u matrix in equation 6 is replaced by v)

$$\boldsymbol{Av} = \boldsymbol{LUv} = \tilde{\boldsymbol{b}} \tag{16}$$

Therefore, two sets of linear equations are generated as bellow for finding $\boldsymbol{v}$:

$$\boldsymbol{Uv} = \boldsymbol{y} \qquad \boldsymbol{Ly} = \tilde{\boldsymbol{b}} \tag{17}$$

Number of FLOPS for LU-decomposition method is [2]

$$\boldsymbol{N_{LU} = \frac{2}{3}n^3 - 2n^2 = \mathcal{O}(\frac{2}{3}n^3)} \tag{18}$$

It means for a matrix with order of 10^5 number of FLOPS equal to $\frac{2}{3} \cdot 10^{15}$.

D. Error estimation

Equation 19 shows a formula that is used to calculate relative error and compare result of numerical solution with exact solution:

$$\epsilon_i = \log_{10}(|\frac{v_i - u_i}{u_i}|) \tag{19}$$

The error should be calculated as a function of step length h (grid points n). Then, its max value is plotted as a function of h, and this provides the maximum produced error by numerical solution.

III. RESULTS AND DISCUSSION

A. Numerical and analytical comparison

General numerical solution and analytical solutions are both computed for matrices of the size 10×10, 100×100, 1000×1000 then, results of $n = 10$ and $n = 100$ are plotted as figure 1 and figure 2 respectively. By comparing figure 1 and figure 2 it is realized that for large n, results of numerical solution is getting closer to exact solution and in contrary, by decreasing n, deviation of numerical from exact solution increases.

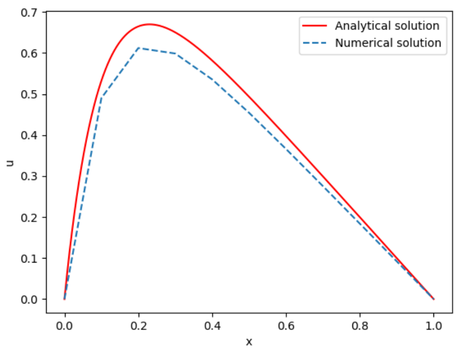

FIG. (1) Comparison of numerical and exact solution for n=10

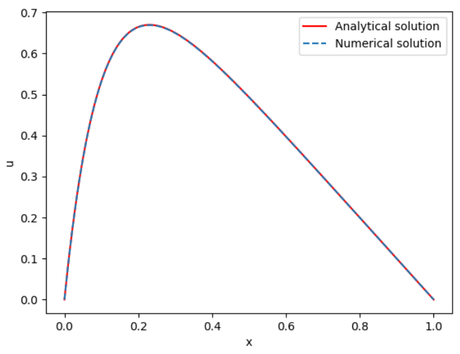

FIG. (2) Comparison of numerical and exact solution for n=100

B. Error analysis

Relative error was calculated according to equation (19) for $n = 10, 10^2, 10^3, 10^4, 10^5, 10^6$ and 10^7 then, maximum relative errors with respect to step size is plotted as figure 3. As can be seen, by increasing n, error decreases up to about $n = 10^5$, then rises with higher slope. When the value of n is further increased, the relative error will become larger. This increase in error for values of $n > 10^5$ is due to the accumulation of round-off errors. When these round-off errors accumulate, they will contribute largely to giving an inaccurate numerical approximation.

C. Run time analysis

To test the efficiency of different methods program run time for general tridiagonal, specific tridiagonal and LU decomposition algorithm is calculated and summarized in table I. As can be seen, compared with general algorithm, the computation took less time with specific algorithm. Furthermore, compared with LU decomposition-based solver, the tridiagonal Gaussian

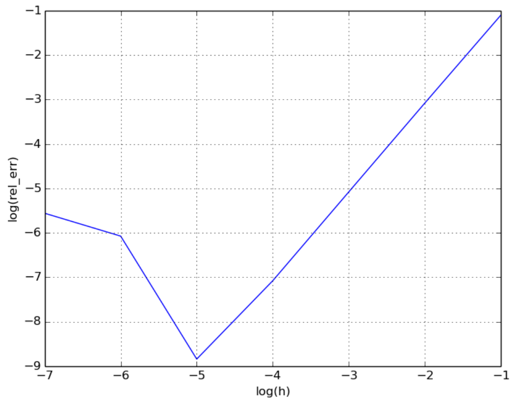

FIG. (3) Maximum relative value

TABLE (I) Run time of different algorithm

n	General[s]	Specific[s]	LU
10	0.0000231	0.0000147	0.00049471
100	0.0002930	0.0001308	0.00042295
1000	0.0019359	0.0011398	0.02385902
10000	0.0281178	0.0116548	16.0864861
100000	0.2144536	0.1143908	n/a
1000000	2.2559521	1.2527589	n/a
10000000	20.312860	12.374752	n/a

methods uses significantly less time for all values of n. In addition, for n > 10000, run time for LU is not accessible because the computer did not have enough memory. However, general technique still works and just spent around 20 seconds for calculations.

IV. CONCLUSION

Overall,specific algorithm is faster than general algorithm and LU decomposition is expensive solution in terms of time and required memory capacity. Furthermore, increasing n lead to accumulation in error.

[1] Hjorth-Jensen, M. (2015). *Computational Physics - Lecture Notes 2018.* University of Oslo

[2] Golub, G.H., van Loan, C.F. (1996). *Matrix Computations* (3rd ed.), Baltimore: John Hopkins.