FYS3150: Project 1

Henrik Haugerud Carlsen and Martin Moen Carstensen

September 9, 2019

Abstract

Often physical systems can be described by linear second-order differential equations, for example the Poisson equation used in electromagnetism. The aim of this report is to rewrite the one-dimensional Poisson equation as a set of linear equations and solve it numerically using algorithms based on the Gaussian elimination method and LU decomposition. The number of FLOPS used by the algorithms and their relative error will be calculated. The specialized algoritm, where the elements below and above the matrix diagonal, was found to be the fastest method for solving the Poisson equation numerically. Also the algorithms relative error was smallest when the steplenght used in the computatitons was of magnitude 10^{-6}.

1 Introduction

The electrostatc potential, $\vec{\phi}$, generated from a localized charge distribution, $\rho(\vec{r})$, in three dimensions can be described by Poissons equation as follows:

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

This can be simplified by utilising the spherical symmetry, which tells us that the potential is constant at at given radius away from the charge distribution independent of change in the polar and/or the asimuttal angle. We can then rewrite the expression to a one-dimensional equation:

$$\frac{1}{r^2}\frac{d}{dr}(r^2\frac{d\vec{\phi}}{dr}) = -4\pi\rho(\vec{r}). \tag{2}$$

By substituting $\vec{\phi} = \frac{\phi(r)}{r}$, we get:

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

and by letting $\phi \rightarrow u$ and $r \rightarrow x$ we get an expression of the general Poisson equation as

$$-u''(x) = f(x) \; . \tag{4}$$

Where $x \in (0,1)$. This can be translated to a set of linear equations. Our problem also contains the boundary conditions $u(0) = u(1) = 0$. Discretizing this equation we get the well known expression for the double derivative:

$$-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i \tag{5}$$

where $i = 1, 2, ..., n$.

To be able to use the Gaussian elimination method an LU decomposition to solve the problem we first have to show that the set of linear equations can be rewritten as

$$A\vec{v} = \vec{b'} \; , \tag{6}$$

where A is an $n \times n$ tridiagonal matrix. The elements of $\vec{b'}$ is $b'_i = h^2 f_i$. First we start by writing out the expression with $i = 1$, getting:

$$-v_2 + v_0 - 2v_i = h^2 f_1 = b'_1 \; . \tag{7}$$

Which translates to

$$-u(2) + u(0) - 2u(1) = h^2 f(1) \tag{8}$$

Inserting the boundary condition $v(0) = v(1) = 0$ we end up with

$$-u(2) = h^2 f(1) \; . \tag{9}$$

This does not correspond with the matrix A given by the problem set.

If we do the same for $i = 2$ we end up with

$$-u(3) - 2u(2) = b' - 2 \; . \tag{10}$$

And for $i = 3$ we get

$$-u(4) + u(2) - 2u(3) = h^2 f(2) \; , \tag{11}$$

and for $i = 4$ we get

$$-u(5) + u(3) - 2u(4) = h^2 f(3) \; , \tag{12}$$

and so on. This leaves us at a general expression for the rows in the matrix as:

$$-u(n) - 2u(n-1) + u(n-2) = b'_n \; . \tag{13}$$

If we insert this for every row in the matrix we end up with a matrix of the from $A\vec{v} = \vec{b'}$.

2 Method

2.1 General algorithm

To solve a set of linear equations of the form $A\vec{v} = \vec{b'}$ an algorithm is developed from the Forward and backward Euler methods.

$$A = \begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots \\ a_1 & b_2 & c_2 & \dots & \dots & \dots \\ & a_2 & b_3 & c_3 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ & & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & & a_{n-1} & b_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \dots \\ \dots \\ \dots \\ f_n \end{bmatrix} .$$

Our matrix is a tridiagonal one so the first step in the algorithm focuses on making the matrix an upper diagonal matrix, in other words eliminating the elements, a_i, below the diagonal. To accomplish this every element b_i has to be mulitiplied by a factor which makes it so that when its is subtracted from the a_i elements the new element is zero. The algorithm is known as the Forward substitution method:

$$b'_i = b_i - \frac{a_{i-1}c_{i-1}}{b'_{i-1}} , \tag{14}$$

with the corresponding equation for the updated f_i's:

$$f'_i = f_i - \frac{a_{i-1}f'_{i-1}}{b'_{i-1}} . \tag{15}$$

The updated matrix then becomes an upper triangular matrix:

$$A = \begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots \\ 0 & b'_2 & c_2 & \dots & \dots & \dots \\ & 0 & b'_3 & c_3 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ & & & 0 & b'_{n-1} & c_{n-1} \\ & & & & 0 & b'_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f'_2 \\ \dots \\ \dots \\ \dots \\ f'_n \end{bmatrix} .$$

Next step is to perform backward substitution which gives us the final diagonal elements:

$$u_i = (f'_i - c_i u_{i+1})/b'_i . \tag{16}$$

Written i pseudocode this method will look like:

```
for $i$ in $2, 3, ..., n$:
        $b'_i = b_i - \frac{a_{i-1}c_{i-1}}{b'_{i-1}}$
        $f'_i = f_i - \frac{a_{i-1}f'_{i-1}}{b'_{i-1}}$
```

$$u_i = \frac{f'_i - c_i u_{i+1}}{b'_i}$$

This loop ends up using 9n FLOPS.

2.2 specialized algorithm

The previous algorithm was made for the general case where knowledge of the elements where more limited. For the case where we know that all the elements $a_i = -1$ and $c_i = -1$ we can write a more effective algorithm to solve the problem. If we write out the calculations we need to find the b'_i elements we get

$$b'_1 = 2$$
$$b'2 = 2 - \frac{1}{2} = \frac{3}{2}$$
$$b'_3 = 2 - \frac{2}{3} = \frac{4}{3}$$
$$b'_4 = 2 - 3\frac{3}{4} = \frac{5}{4}$$

We recognize the pattern as

$$b'_i = \frac{i+1}{i} \; . \tag{17}$$

Because we know the constants a_i and c_i the updated values of f_i and u_i becomes:

$$f'_i = f_i + \frac{f_{i-1}}{b_{i-1}} \tag{18}$$

$$u_i = \frac{f'_i + u_{i+1}}{b'_i} \tag{19}$$

So the pseudocode becomes:
for i in $2, 3, ..., n$:

$$b'_i = \frac{i+1}{i} \; .$$
$$f'_i = f_i + \frac{f_{i-1}}{b_{i-1}}$$
$$u_i = \frac{f'_i - c_i u_{i+1}}{b'_i}$$

This method only uses 4N FLOPS, so it the special algorithm is faster than the general one, as expected.

2.3 LU decomposition

We will compare the previously discussed methods with LU decomposition. LU decomposition takes the matrix, A, and writes it as a matrixproduct of a lower triangular matrix, L, and an upper triangular matrix, U.

$$L = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots \\ l_{21} & 1 & 0 & \dots & \dots & \dots \\ l_{31} & l_{32} & 1 & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & 1 & 0 \\ l_{i1} & \dots & \dots & \dots & l_{ij-1} & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & \dots & \dots & u_{1j} \\ 0 & u_{22} & \dots & \dots & \dots & \dots \\ \dots & 0 & u_{33} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & 0 & u_{ij} \end{bmatrix}$$

The elements can be found by first calculating the elements $u_{1j} = a_{1j}$, then calculating the elements u_{ij}, where $i = 2, 3, ..., j - 1$, with the following equation:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \; . \tag{20}$$

Then find the diagonal elements from

$$u_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk} u_{kj} \; . \tag{21}$$

Lastly calculate the remaining elements:

$$l_{ij} = \frac{1}{u_{jj}} (a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \; . \tag{22}$$

When this is done the decomposed system can be solved by first solving $L\vec{z} = \vec{f}$, then solving $U\vec{x} = \vec{z}$. Here we set $U\vec{x} = \vec{z}$. What we end up with is a system which can be solved by for loops without more decomposition.

For the code used in this paper to used in this paper for LU decomposition see [1].

2.4 Relative error

The relative error, ϵ_i for each iteration, is given by

$$\epsilon_i = log_{10}(|\frac{v_i - u_i}{u_i}|) \ . \tag{23}$$

3 Results

Using both our general and specialized algorithm we can compute the numerical solution to our matrix equation. This was done for an n ranging from $n = 10^1$ up to $n = 10^7$. The plots for $n = 10, 10^3$ and 10^5 is show subsequently.

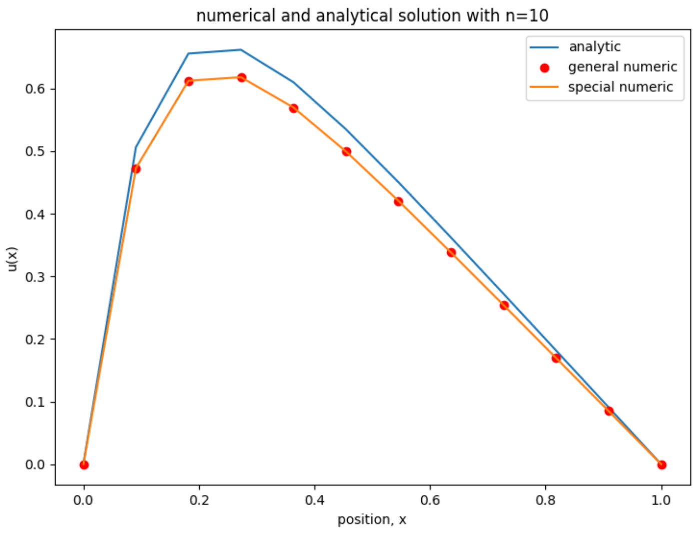

Figure 1: Analytical, general and specialized solution to the poisson equation for $n = 10^1$

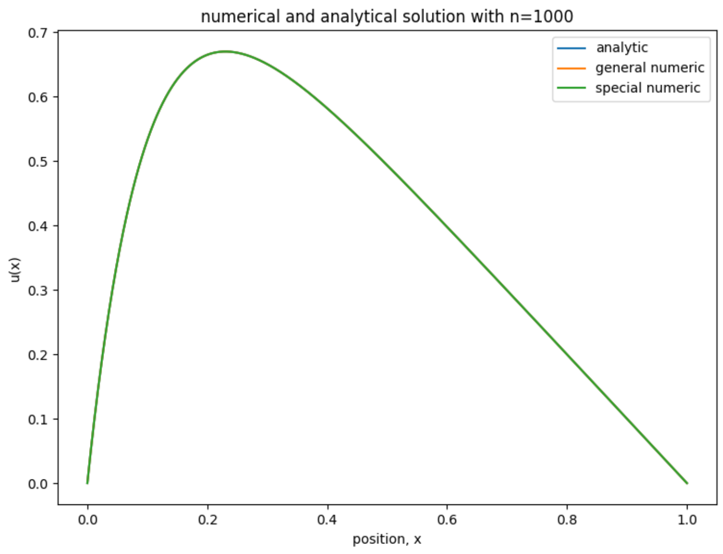

Figure 2: Analytical, general and specialized solution to the poisson equation for $n = 10^3$

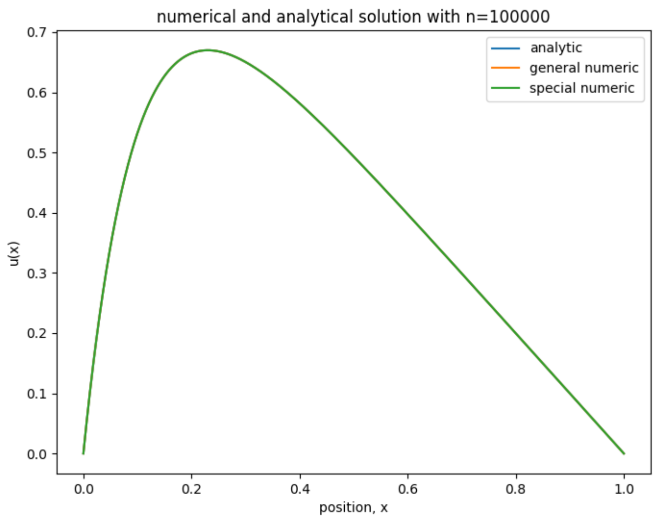

Figure 3: Analytical, general and specialized solution to the poisson equation for $n = 10^5$

At the same time we computed the relative error of the specialized algorithm against the analytical solution. This was done to get a relation between the max relative error of the calculations and the steplengths, h.

For this, equation **??** was used. By plotting the maximum error of each steplength as a function of the briggsian logarithm of h we got the plot

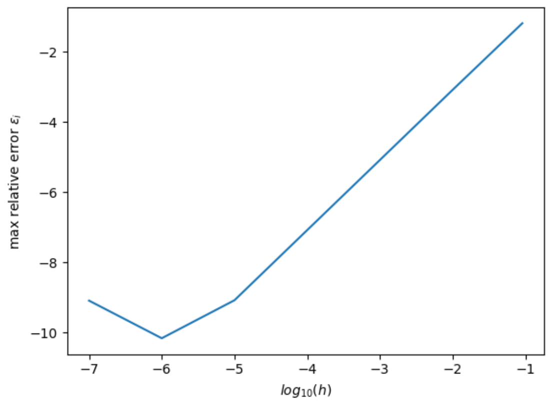

Figure 4: The max relative error plotted as a function of $log_{10}(h)$ is shown.

The speed of computations is shown in table **??**. There was also used an even more general LU decomposition algorithm that was compared to the time consumption of the general and special algorithm. The time spent is presented in a tabular form here:

Table 1: time spent for general, specialized and LU decomposition for the different n

n	General (s)	Specialized (s)	LU decomp. (s)
10^1	$< 1 \cdot 10^{-4}$	$< 1 \cdot 10^{-4}$	$< 1 \cdot 10^{-3}$
10^2	$1 \cdot 10^{-3}$	$< 1 \cdot 10^{-3}$	0.14
10^3	$5 \cdot 10^{-3}$	$4 \cdot 10^{-3}$	154
10^4	$5.7 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	-
10^5	$3.9 \cdot 10^{-1}$	$1.9 \cdot 10^{-1}$	-
10^6	3.4	1.6	-
10^7	31.8	15.5	-

4 Discussion

In terms of accuracy we see from figure **??**, **??**, **??** that there is no difference between the general and special algorithm for solving the Poisson equation. Their only difference was their speed. The special algorithm being the superior computing method, see table **??**. The general and special algorithm used a lot less time than the LU decomposition algorithm. This was expected before the computations was done because of the FLOP count difference between the methods. Because of the larger timeconsumption of the LU decomposition method no specific value was found, but for a matrix of size $10^4 \times 10^4$ was guesstimated to be about 100.000 seconds because of the increase in timeconsumption between the smaller matrices. They had a timeconsumption increas of a factor 10^3 as n increased by 1. Following this logic we should expect to encounter a memory error when $n \geq 5$. The max relative error was examined and was found to be decreasing when the steplength went from a magnitude of 10^{-7} to 10^{-6} where is was at it's lowest. Looking at table **??**, the relative error almost increases linearly with the subsequent changes in steplength magnitude. This correlates to an exponential increase in relative error if the error was plottet against the steplength and not the briggsian logarithm of the steplength.

5 Conclusion

In this paper we have derived both a general and special algorithm for computing the solution of the Poisson equation. The algorithms where compared against the LU decomposition method for solving the system of equations, and the maximal relative error was plotted as a function of the steplength used in the specialized algorithm.
The fastest algorithm for solving the Poisson equation was found to be the specialized algorithm. The accuracy of the general and special algorithm where the same. Optimally the steplenght for the specialized algorithm should be of a magnitude 10^{-6} to produce the minimal relative error. The relative error increases exponentially as the magnitude of the steplenght goes from 10^{-6} to 10^{-1}.

6 References

[1]: Code by Morten Hjorth-Jensen (the first 200 lines are used):
`https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Programs/LecturePrograms/programs/pythonLibrary/computationalLib.py?fbclid=IwAR0hMqS-TOOL7oBTiWEY3jGUNgwFayE9ej4SN37zovWCsiJGrQUPqxSwXKk`

[2]: Lecture notes FYS3150, Fall 2015, by Morten Hjorth-Jensen. Chapter 6.
`https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf`