

# FYS4150 - Project 1

HEINE H. NESS & SINDRE R. BILDEN\*

University of Oslo

h.h.ness@fys.uio.no ; s.r.bilden@fys.uio.no  
github.com/sindrerb/FYS4150-Collaboration

September 17, 2016

## Abstract

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.*

## I. INTRODUCTION

**T**his project will examine different techniques for approximating the solution to a differential equation where a continuous function is known. The equation describes an electrostatic potential  $\Phi$  generated by a localized charge density  $\rho(\vec{r})$  and is usually described - in three dimensions - by:

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

If  $\rho(\vec{r})$  is spherical symmetric, eq. 1 may be written in a one-dimensional manner by substituting  $\phi(r) = r\Phi(r)$ :

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

By rewriting eq. 2 to a general form it reads:

$$-u''(x) = f(x) \quad (3)$$

In this specific case, the Poisson equation is solved by *Gaussian elimination* of a set of linear equations, both in a general manner and an optimized way of a specific matrix. The optimized method is later compared with another general method called *LU-decomposition*.

---

\*A thank you or further information

## II. METHODS

The methods used in this projects are the following:

- Dirichlet boundary conditions
- Numerical derivation
- Thomas algorithm
- LU-decomposition
- Numerical error analysis

### i. Dirichlet boundary condition

Dirichlet boundary conditions - also referred to as fixed boundary condition - specifies the value of a given function on a surface  $T = f(r, t)$ . In a one-dimensional problem it translates to defining an interval of  $x$  -  $x \in [x_{min}, x_{max}]$  - and the function values  $f(x_{min}) = f_{low}$  and  $f(x_{max}) = f_{high}$  at the edges of the interval.

### ii. Numerical derivation

The derivative of a discrete function may be found by numerical derivation. The principle of numerical derivation is a result of Taylor expansion. By expanding a function

from a point  $x$  with a step  $h$ , two equations form depending on the direction of the step:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) \dots \quad (4)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) \dots \quad (5)$$

By adding eq. 5 to eq. 4, a approximation for the second derivative is achieved.

$$f'' = \frac{f_+ - 2f + f_-}{h^2} + \frac{h^4}{6h^2}f^{IV} \quad (6)$$

Where  $f_+ = f(x+h)$ ,  $f = f(x)$ ,  $f_- = f(x-h)$  and  $f^{IV}$  is the fourth derivative of  $f(x)$ . By truncating the series at the fourth derivative a small mathematical error appears in the order of  $h^2$ . If a discrete funtion is introduced where  $f_i = f(x_i) = f(x_0 + ih)$ , eq. 6 may be rewritten to an algorithm for the numnerical second derivative.

$$f''_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \quad (7)$$

In eq. 7 the mathematical error  $\mathcal{O}(h^2)$  is neglected.

### iii. Thomas algorithm

Thomas algorithm is a variant of Gaussian elimination used on tridiagonal matrices [? ], where Gaussian elimination is a method using  $\mathcal{O}(n^3)$  floating point operations (FLOPS) for solving a set of  $n$  linear equations with  $n$  unknown variables  $x_i$   $i = 0, 1, \dots, n-1$ . Thomas algorithm is usefull if the equations is on the form :

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = y_i$$

Resulting in a matrix equation  $Ax = y$  where  $A$  and  $y$  is known.

$$\begin{bmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & a_3 & b_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \dots & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_n \end{bmatrix}$$

Both Thomas algorithm and Gaussian elimination is divided into two main parts, forward and backward substitution.

### Forward substitution

The forward substitution is focusing on reducing the number of elements in a column to a minimum. In other words, row reduction is first used on the matrix  $A$  to eliminate all elements  $a_i$ . Turning the matrix equation to  $Bx = \hat{y}$ :

$$\begin{bmatrix} b_1 & c_1 & 0 & \dots & 0 \\ 0 & \tilde{b}_2 & c_2 & \ddots & \vdots \\ 0 & 0 & \tilde{b}_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \dots & 0 & 0 & \tilde{b}_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \\ \tilde{y}_4 \\ \vdots \\ \tilde{y}_n \end{bmatrix}$$

where  $\tilde{b}$  and  $\tilde{y}$  is affected by the row reduction. The new set of linear equations is the basis for backward substitution.

### Backward substitution

The concept of backwars substitution is to solve the the set of equations from bottom to top. The bottom  $x$  can be expressed as  $x_n = \frac{\tilde{y}_n}{\tilde{b}_n}$  and may be used to solve the equation above. In the end, all elements of  $x$  is known. In total, the Thomas algorithm is using  $9n$  FLOPS. If the matrix elements is constant through every diagonal, the number of FLOPS may be reduced to  $4n$ .

### iv. LU-decompostition

LU decomposition (LU) also known as LU-, Crout or Dolittle factorisation [? ][?] is a common way to solve matrix problems numerically. It is used to find important matrix properties such as the determinant.

Most importantly it reduces the number of FLOPS used to solve linear algebra problems, from  $\mathcal{O}(n^3)$  FLOPS by a regular Gaussian elimination to  $\mathcal{O}(n^2)$  by LU-decomposition.

LU-decomposition as the name suggests decomposes a matrix  $\hat{A}$  into two matrices  $\hat{L}$  and  $\hat{U}$  where  $\hat{L}$  is a lower triangular matrix with ones along its diagonal and  $\hat{U}$  is an upper triangular matrix. For this to be possible  $\hat{A}$  needs

to be a square, invertible matrix and its leading principal minors must be nonzero.

### v. Numerical error estimate

Since numerical methods are used, the result is only an approximation to the analytical answer. Therefore it is interesting to see how good the numerical method is by comparing the approximation to an already known solution. A good way to do this is to look at the relative error  $\varepsilon_r$  between the approximation and the exact solution. At small step lengths  $h$ , the approximation is often close to the exact solution if the theory is correct. Therefore it is convenient too look at the logarithm of the relative error to differentiate methods of nearly equal accuracy. An appropriate formula is:

$$\varepsilon_i = \log_{10} \left( \left| \frac{v_i - u_i}{u_i} \right| \right) \quad (8)$$

Where  $v_i$  is the numerical value and  $u_i$  is the analytical value at step  $i$ .

## III. IMPLEMENTATION

By discretizing the simplified and generalized Poisson equation (eq. 6) in steps of  $h$ , we may use eq. 7 to write eq. 9 where  $v_i$  is the discrete values of the continuum  $u(x)$ .

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

The equation may be rewritten into a linear equation

$$av_{i-1} + bv_i + cv_{i+1} = \tilde{b}_i$$

where  $a = -1$ ,  $b = 2$ ,  $c = -1$  and  $\tilde{b}_i = f_i h^2$ . By introducing dirichlet boundary conditions,  $v_0 = v_{n+1} = 0$ , the linear equations may be written in terms of matrix multiplication without loss of information. This results in  $Av = \tilde{b}$ :

$$\begin{bmatrix} b & c & 0 & 0 & \cdots & 0 \\ a & b & c & 0 & \cdots & 0 \\ 0 & a & b & c & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & a & b \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_0 \\ \tilde{b}_1 \\ \vdots \\ \tilde{b}_{n-2} \\ \tilde{b}_{n-1} \end{bmatrix} \quad (10)$$

The problem is now solvable by both LU-decomposition and Thomas algorithm. Since matrix  $A$  has a constant value along its diagonals, an optimized Thomas algorithm was constructed to this specific matrix. All codes are tested with a reference function  $f(x)$  with the known solution  $u(x)$  where

$$f(x) = 100 \exp[-10x]$$

$$u(x) = 1 - \exp[-10]x - 10 \exp[-10x]$$

For all programs, divided in task  $b$ ,  $c$ ,  $d$  and  $e$  see the github repository:

[github.com/sindrerb/FYS4150-Collaboration](https://github.com/sindrerb/FYS4150-Collaboration)

## IV. RESULTS AND DISCUSSION

### i. LU-decomposition

Solving eq. 10 using LU-decomposition is a time consuming process and is often a non-satisfying method for square matrices of size above  $10^4$  due to limits in memory. Table 1 shows the computation time for a set of matrices of size  $N$ .

$N$	1e1	1e2	1e3	1e4
$t$ [s]	2.78e-4	4.62e-3	8.198e-1	8.614e2

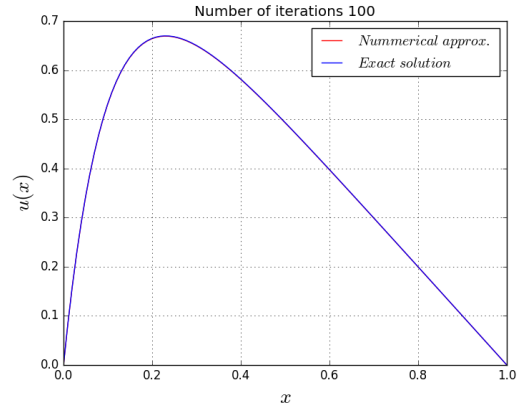
**Table 1:** Table showing computation time  $t$  in seconds for LU-decompositon of matrices of size  $N$ .

### ii. General Thomas algorithm

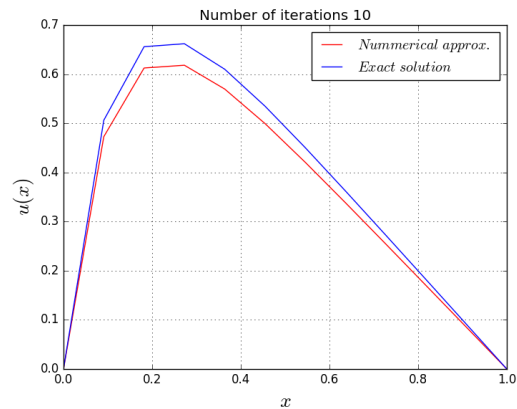
Solving eq. 10 by the more suited Thomas algorithm reduced the computation time significantly compared to the LU-decomposition. Thomas method gave a - by eye - satisfying approximation at 100 iterations, as seen in Figure 1. Although - by investigating the numerical error analysis - the increased number of iterations shows a significant increase of accuracy up to  $N = 10^6$  iterations, at higher number of iterations the accuracy seems to drop as seen i Table 2. The algorithm seems to have a tendency to underestimate the derivative and will approach the exact solution from below at a low number of iterations, as seen in Figure 2.

Iterations	Relative error
1e1	-1.17970
1e2	-3.08804
1e3	-5.08005
1e4	-7.07927
1e5	-9.07909
1e6	-10.7943
1e7	-9.54215

**Table 2:** Table showing the relative error based on eq. 8 for a set of iterations.



**Figure 1:** Plot of the numnerical approximation of  $u(x)$  - in red - with 100 iterations by Thomas algorithm, compared to the exact solution  $u(x)$  in blue.



**Figure 2:** Plot of the numnerical approximation of  $u(x)$  - in red - with 10 iterations by Thomas algorithm, compared to the exact solution  $u(x)$  in blue.

### iii. Specialized Thomas algorithm

Solving eq. 10 with a specialized Thomas algorithm gave - as expected - the same results as the general algorithm, but reduced the number FLOPS from  $9n$  in the general algorithm to  $4n$  in the specialized algorithm. The reduction in number of FLOPS shortened the computation time. A difference in the computation time is hard to spot at a low number of iterations, but a higher number of iterations reveals a difference. A comparison of the timeusage by the two algorithms is found in Table 3.

Iterations	Timeusage [s]	
	General	Special
1e1	1.0e-6	1.0E-6
1e2	9.0e-6	5.0e-6
1e3	5.1e-5	4.5e-5
1e4	7.66e-4	7.95e-4
1e5	5.021e-3	3.075e-3
1e6	2.6094e-2	2.277e-2
1e7	2.7316e-1	2.26463e-1

**Table 3:** Table showing timeusage - in seconds - by the special and the general Thomson algorithm for a set of iterations.

## V. CONCLUSION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

