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 15, 2016

Abstract

Lorem ipsum dolor sit amet, consectetuer 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, consectetuer 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

This project will examine different techniques for approximating the solution to a differential equation where a continious function is known. The equation describes an electrostatic potential Φ generated by a localized charge density $\rho(\vec{r})$ and is usualy described - in three dimentions - by:

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

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

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

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

$$-u''(x) = f(x) \tag{3}$$

In this spesific 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 spesific 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
- Nummerical derivation
- Gaussian elimination
- LU-decomposition

i. Dirichlet boundary condition

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

ii. Nummerical derivarion

The derivative of a discrete funtion may be found by nummerical derivation. The principle of nummerical 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:

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

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

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

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

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 - $\mathcal{O}$ - appears in the order of h^2. If a discrete funtion is introduced where $f_i = f(x_i) = f(c_0 + ih)$, eq. 6 may be rewritten to an algorithm for the nummerical second derivative.

$$f_i'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \tag{7}$$

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

iii. Gaussian elimination

Gaussian elimination is a method for simplifying a set of N linear equations with n unknown variables x_i $i = 0, 1, \ldots, n-1$:

$$\begin{aligned} a_{00}x_0 + a_{01}x_1 + a_{02}x_2 &= y_0 \\ a_{10}x_0 + a_{11}x_1 + a_{12}x_2 &= y_1 \\ a_{20}x_0 + a_{21}x_1 + a_{22}x_2 &= y_2 \end{aligned}$$

It is easly visualized through a matrix notation of $Ax = y$ where A and y is known.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \tag{8}$$

Gaussian elimination is often divided into two main parts, forward and backward substitution.

Forward substitution

The forward substitution is focusing on reducing the number of variables in the set of linear equations to a minimum. In other words, row reduction is used on the matrix A to eliminate all elements a_{i1} where $i < 1$. Turning the matrix equation to $Bx = \hat{y}$:

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} \\ 0 & b_{11} & b_{12} \\ 0 & b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \tag{9}$$

where $\hat{y}$ is affected by the row reduction. The process is repeated until matrix A is transformed to an upper triangular matrix A'

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ 0 & a_{11} & a_{12} \\ 0 & 0 & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} \tag{10}$$

This 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 bsnto the equation above. In the end, all elements of x is known. Gaussian elimination optimized for a tridiagonal matrix is called Thomas algoritm. [kilde?]

iv. LU-decompostition

v. Numerical error estimate

Since we are using numerical derivation this gives us an approximation to the analytical answer. Therfore it is interesting to see how good our approximation is. A good way to do this is to look at the difference between the two.

At small step lenghts (h) we will have a good approximation if the theori is correct and therfore it is convenient too look at the logaritme of the difference we also use the absolute walue since we dont care if we are over or under the analytical answer. To avoid the logaritme of zero we also devide by the analytical value.

So we get the relative error ε and for each step ve get:

$$\varepsilon_i = log\left(\left|\frac{u_i - v_i}{v_i}\right|\right) \tag{11}$$

Here u_i is the numerical value and v_i is the analytical value at step i. And log is the base 10 logaritme.

III. Implementation

By discretizing the simplified and generalized Poiison equation (eq. 6) in steps of h, we may approximate eq. 6 with eq. 12 where v_i is the discrete values of the continium $u(x)$.

$$-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} = f_i \tag{12}$$

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 $Av = \tilde{b}$:

$$\begin{bmatrix} b & c & 0 & 0 & \cdots & 0 \\ a & b & c & 0 & \cdots & 0 \\ 0 & a & b & c & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & c \\ 0 & 0 & 0 & 0 & a & b \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_0 \\ \tilde{b}_1 \\ \vdots \\ \tilde{b}_{n-2} \\ \tilde{b}_{n-1} \end{bmatrix} \tag{13}$$

The problem is now solvable by both Thomas algoritm and LU-decomposition. Since matrix A has a constant value along it's diagonals, an optimized Thomas algoritm was constructed to this spesific 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 and d see the github repository:

github.com/sindrerb/FYS4150-Collaboration

IV. Results

i. LU-decomposition

ii. General Thomas algoritm

Solving eq. 13 using Thomas algoritm gave a - by eye - satisfying approximation with 100 iterations, as seen in Figure 1. The algorithm seems to has a tendency to underestimate the derivative and will approach exact solution from below, as seen in Figure 2.

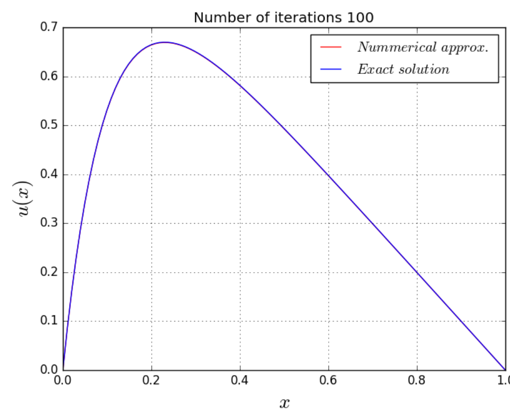

Figure 1: *Plot of the nummerical approximation of $u(x)$ - in red - with 100 iterations by Thomas algoritm, compared to the exact solution $u(x)$ in blue.*

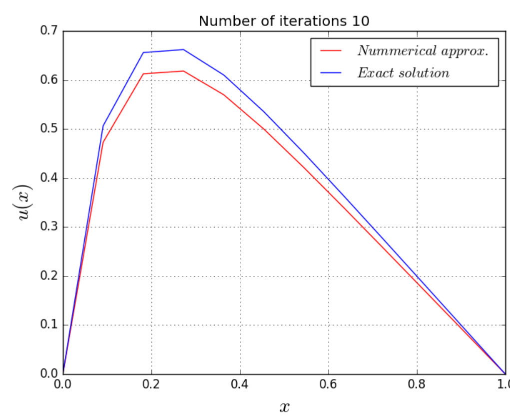

Figure 2: *Plot of the nummerical approximation of $u(x)$ - in red - with 10 iterations by Thomas algoritm, compared to the exact solution $u(x)$ in blue.*

iii. Specialized Thomas algoritm

Solving eq. 13 with a specialized Thomas algorithm gave the same results as the general algoritm, but reduced the number floating operations from $9n$ in the general algoritm to $4n$ in the specialized algoritm. The reduction in number of floating operations shorted the comutation time. A difference is hard to spot at a low numer of iterations, but higher number of iterations reveals a difference. A comparation of the timeusage by the two algoritms is found in Table 1.

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

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

Table 2: *Table showing the relative error for a set op iterations.*