Solving the 1D-Poisson Equation Numerically

Kristian Gjestad Vangsnes

Department of physics, University of Oslo

Date: 9. September 2019

Abstract

abstract

1 Introduction

In this project we will study numerical algorithm which solves the differential equation Poisson's equation, this is the equation which modulates the change of a electric potential.

Two algorithms will be based on Gaussian-elimination and is programed on a lower level using c++, while the third algorithm is the famous LU-decomposition and will be computed using the linear algebra library Armadillo.

The focus of this report is to compare the efficiency and precision of the algorithms.

2 Theory

The one-dimentional Poisson equation with Dirichlet boundary conditions reads

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

We will use the source term $f(x) = 100e^{-10x}$. This gives the particular solution $u(x) = 1 - (1 - e^{-10})x - e^{-10x}$. We will compare our numerical solution to this exact solution.

3 Numerical methods

In order to model this equation in a computer we need to define the discretized approximated solution to $u(x)$ as $v_i = v(x_i)$ where $x_i = x_0 + ih = ih$ since

$x_0 = 0$. We let $x_{n+1} = 1$, this means $h = \frac{1}{n+1}$. From Taylor expanding $u(x \pm h)$ we get

$$u(x \pm h) = u(x) \pm hu'(x) + \frac{h^2}{2!}u''(x) \pm O(h^3)$$

We see that $u''(x) = \frac{u(x+h)+u(x-h)-2u(x)}{h^2} - O(h^4)$. Hence for the approximated solution we get that

$$-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i \quad \text{for i} = 1, ..., n$$

where $f_i = f(x_i)$. If we set $g_i = h^2 f_i$ we get that $2v_i - v_{i+1} - v_{i-1} = g_i$. This is just n equations, given by $i = i, ..., n$. We can write this in matrix form

$$\mathbf{Av} = \mathbf{g},$$

where $\mathbf{A}$ is a $n \times n$ tridiagonal matrix on the form

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & ... & 0 \\ -1 & 2 & -1 & 0 & ... & 0 \\ 0 & -1 & 2 & -1 & 0 & ... \\ \vdots & 0 & \ddots & \ddots & \ddots & ... \\ 0 & ... & ... & -1 & 2 & -1 \\ 0 & ... & ... & 0 & -1 & 2 \end{bmatrix}$$

$\mathbf{v}$ and $\mathbf{g}$ are vectors on the form

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \qquad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}$$

3.1 General algorithm

The general algorithm to solve a set of equation on a tridiagonal form is done by Gaussian elimination, and is called the Thomas algorithm [1]. The algorithm On matrix form the problem is $\mathbf{Av} = \mathbf{g}$. Where $\mathbf{v}$ contains the unknowns v_i. In our case the unknowns are the solution to the Poisson equation at $x_i = ih$, i.e. $v(x_i) = v_i$.

$$\mathbf{A} = \begin{bmatrix} d_1 & b_1 & 0 & \dots & ... & 0 \\ a_1 & d_2 & b_2 & 0 & ... & 0 \\ 0 & a_2 & d_3 & b_3 & 0 & ... \\ \vdots & 0 & \ddots & \ddots & \ddots & ... \\ 0 & ... & ... & a_{n-2} & d_{n-1} & b_{n-1} \\ 0 & ... & ... & 0 & a_{n-1} & d_n \end{bmatrix}, \qquad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix}.$$

We see that the two first equations are

$$d_1 v_1 + b_1 v_2 = g_1 \tag{1}$$

$$a_1 v_1 + d_2 v_2 + b_2 d_3 = g_2 \tag{2}$$

We see that if we multiply eq (1) by $\frac{a_1}{d_1}$ and subtract it from eq(2) we see that eq(2) becomes

$$(d_2 - \frac{a_1 b_1}{d_1}) = g_2 - \frac{a_1 g_1}{d_1}$$

.

This is called forward substitution and generally for a tridiagonal matrix gives us the new diagonal elements

$$\tilde{d}_{i+1} = d_{i+1} - \frac{a_i b_i}{\tilde{d}_i}$$

$$\tilde{g}_{i+1} = g_{i+1} - \frac{a_i \tilde{g}_i}{\tilde{d}_i}$$

Where $\tilde{d_1} = d_1$ and $\tilde{g}_1 = g_1$. Now our problem in on the form $\tilde{\mathbf{A}}\mathbf{v} = \tilde{\mathbf{g}}$ where

$$\tilde{\mathbf{A}} = \begin{bmatrix} \tilde{d_1} & b_1 & 0 & \dots & ... & 0 \\ 0 & \tilde{d_2} & b_2 & 0 & ... & 0 \\ 0 & 0 & \tilde{d_3} & b_3 & 0 & ... \\ \vdots & 0 & \ddots & \ddots & \ddots & ... \\ 0 & ... & ... & 0 & \tilde{d}_{n-1} & b_{n-1} \\ 0 & ... & ... & 0 & 0 & \tilde{d_n} \end{bmatrix}, \qquad \mathbf{g} = \begin{bmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \vdots \\ \tilde{g}_n \end{bmatrix}.$$

To get the problem to a diagonal form we have to perform a backward substitution, this is done component wise by

$$v_{n-i} = \frac{\tilde{g}_{n-i} - b_{n-i} v_{n+1-i}}{\tilde{d}_{n-i}}$$

where $v_n = \frac{\tilde{g}_n}{\tilde{d}_n}$.

When we start to count from 0 to $n-1$ the algorithm goes as follows:

General Algorithm
Forward substitution
for $i = 0, 1, 2, ..., n-1$
$d_{i+1} -= a_i * b_i / d_i$
$g_{i+1} -= a_i g_i / d_i$
End loop
Backward substitution
$v_{n-1} = g_{n-1} / d_{n-1}$
for i=2, 3, ..., n
$v_{n-i} = (g_{n-i} - b_{n-i} v_{n+1-i}) / d_{n-i}$
End loop

The number of floating points operations (FLOPS) performed in this general algorithm is $9n$, $6n$ FLOPS in the forward substitution, $2n$ subtractions, $2n$ multiplications and $2n$ divisions. $3n$ FLOPS in the backward substitution, n subtraction, n multiplication and n division.

3.2 Specialized algorithm

When all the diagonal elements are equal and the off diagonal elements are equal we can make our algorithm more efficient. If we us the matrix **A** for the Poisson equation, we get a even more specialized algorithm. For equal diagonal elements and equal off diagonal elements we get the following algorithm.

Specialized Algorithm for equal diagonal and off diagonal terms
Diagonal term is d and off diagonal term is a.
$d_0 = d$
Let $A = a^2$, we precalculate in order to save memory.
Forward substitution
for $i = 0, 1, 2, ..., n-1$
$d_{i+1} -= A / d_i$ Note
$g_{i+1} -= a \times g_i / d_i$
End loop
$v_{n-1} = g_{n-1} / d_{n-1}$
Backward substitution
for i=2, 3, ..., n
$v_{n-i} = (g_{n-i} - a \times v_{n+1-i}) / d_{n-i}$
End loop

This algorithm does $8n$ FLOPS, it does $5n$ FLOPS in the forward substitution, $2n$ subtractions, $2n$ divisions and n multiplications. In the backward substitution we have $3n$ FLOPS, n subtractions, n multiplications and n divisions.

But we can make a even better algorithm if the diagonal elements are 2 and the off diagonal elements are -1. If that is the case we can use the following algorithm.

Specialized Algorithm for the Poisson problem
Note tha t the diagonal elemets are precalculated
Forward substitution
for $i = 0, 1, 2, ..., n-1$
$g_{i+1} += g_i/d_i$
End loop
Backward substitution
$v_{n-1} = g_{n-1}/d_{n-1}$
for i=2, 3, ..., n
$v_{n-i} = (g_{n-i} + v_{n+1-i})/d_{n-i}$
End loop

We see that since we have precalculated multiple

4 Program structure

5 results

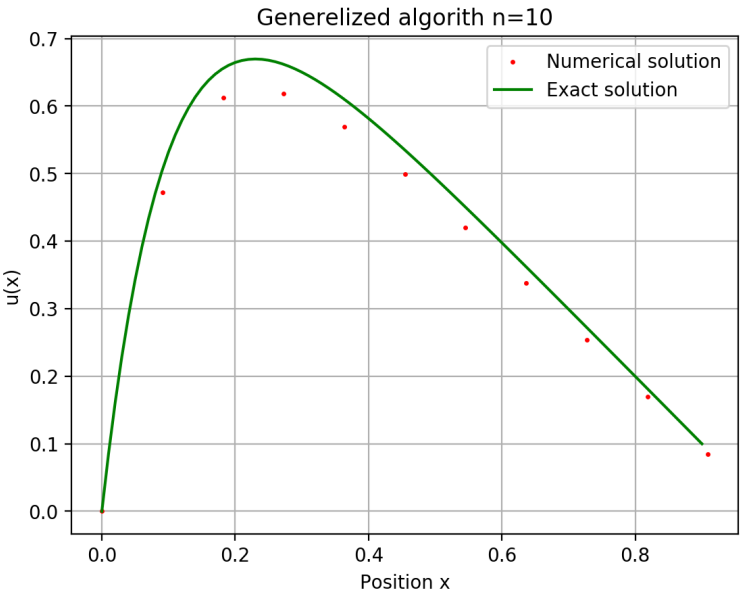
The general algorithm was programed using c++ and the solution for matrix size n=10, 100 and 1000 was compared to the exact solution and plotted using python as shown below.

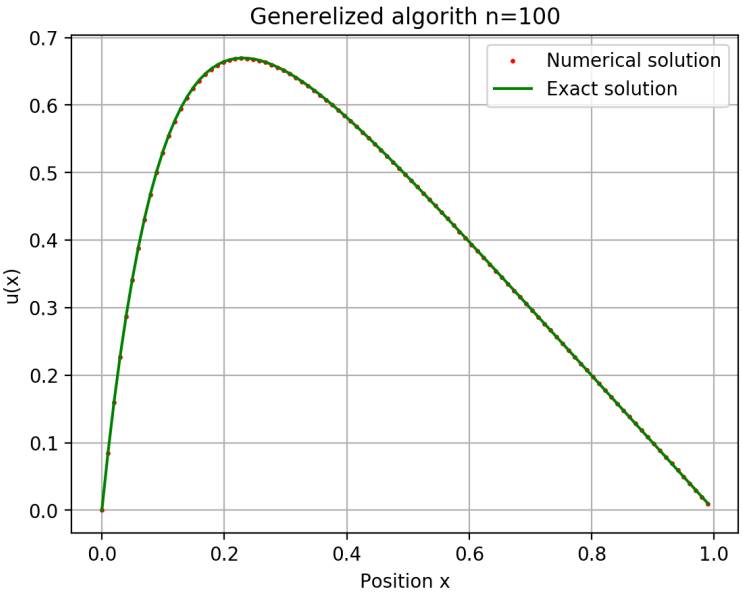
We see that for the general solution $n = 10$ is a bad approximation, this is because the step length is too long, i.e. too few grid points, which makes the approximation of the second derivative bad.

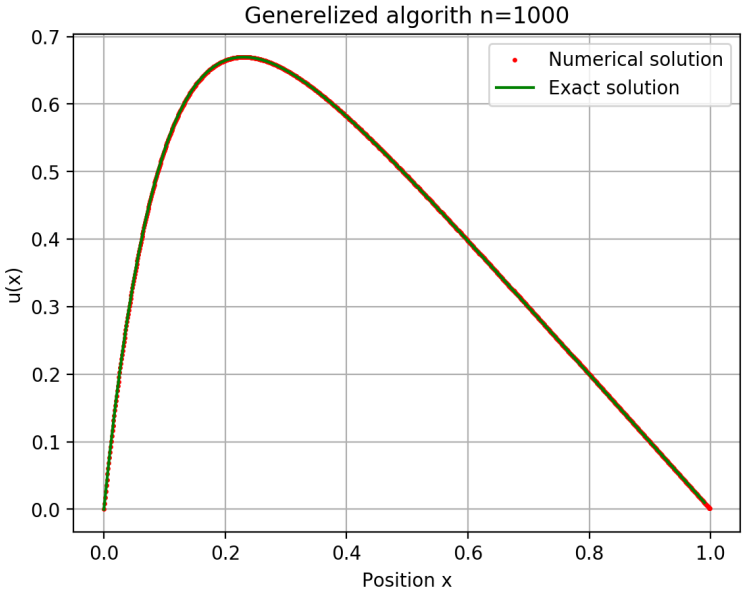
6 Summary

References

[1] L.H. Thomas. "Elliptic Problems in Linear Differential Equations over a Network". In: *Watson Sci. Comput. Lab Report* (1949).


Generelized algorith n=10
Numerical solution
Exact solution
u(x)
Position x
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.2
0.4
0.6
0.8


Generelized algorith n=100
Numerical solution
Exact solution
u(x)
Position x
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.2
0.4
0.6
0.8
1.0


Generelized algorith n=1000
Numerical solution
Exact solution
u(x)
Position x
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
0.2
0.4
0.6
0.8
1.0