Project 1 FYS3150

Furkan Kaya

September 2021

1 Introduction

This is Furkan Kaya's answer to the questions posed in the first project in the course FYS3150: Computational Physics. Project 1 tries to give an introduction into the programming language C++ through solving a given assignment. The overall topic of this project is numerical solution of the one-dimensional Poisson equation. This is a second-order differential equation that shows up in several areas of physics, e.g. electrostatics.

2 Problem 1

The Poisson equation is given by:

$$-\frac{d^2u}{dx^2} = f(x) \tag{1}$$

where f(x) = $100 * exp(-10x)$, x ϵ $[0, 1]$ and the boundary conditions u(0) = 0 and u(1) = 0.

We are to check analytically that an exact solution (1) is given by (2).

$$u(x) = 1 - (1 - exp(-10))x - exp(-10x) \tag{2}$$

So, I do a double derivation to check if (2) is a solution to (1).

$$\frac{du}{dx} = (1 - exp(-10)) + 10 * exp(-10x) \tag{3}$$

$$-\frac{d^2u}{dx^2} = 100 * exp(-10x) \tag{4}$$

3 Problem 2

I did this problem in MATLAB. The entire solution can be found in the code with the plot and data file also added.

```
x = linspace(0,1)

u = 1 - (1 - exp(-10))*x - exp(-10*x)

plot(x,u);
```

```
xlabel('x');
ylabel('u(x)');
title('Analytical solution to the Poisson equation');
```

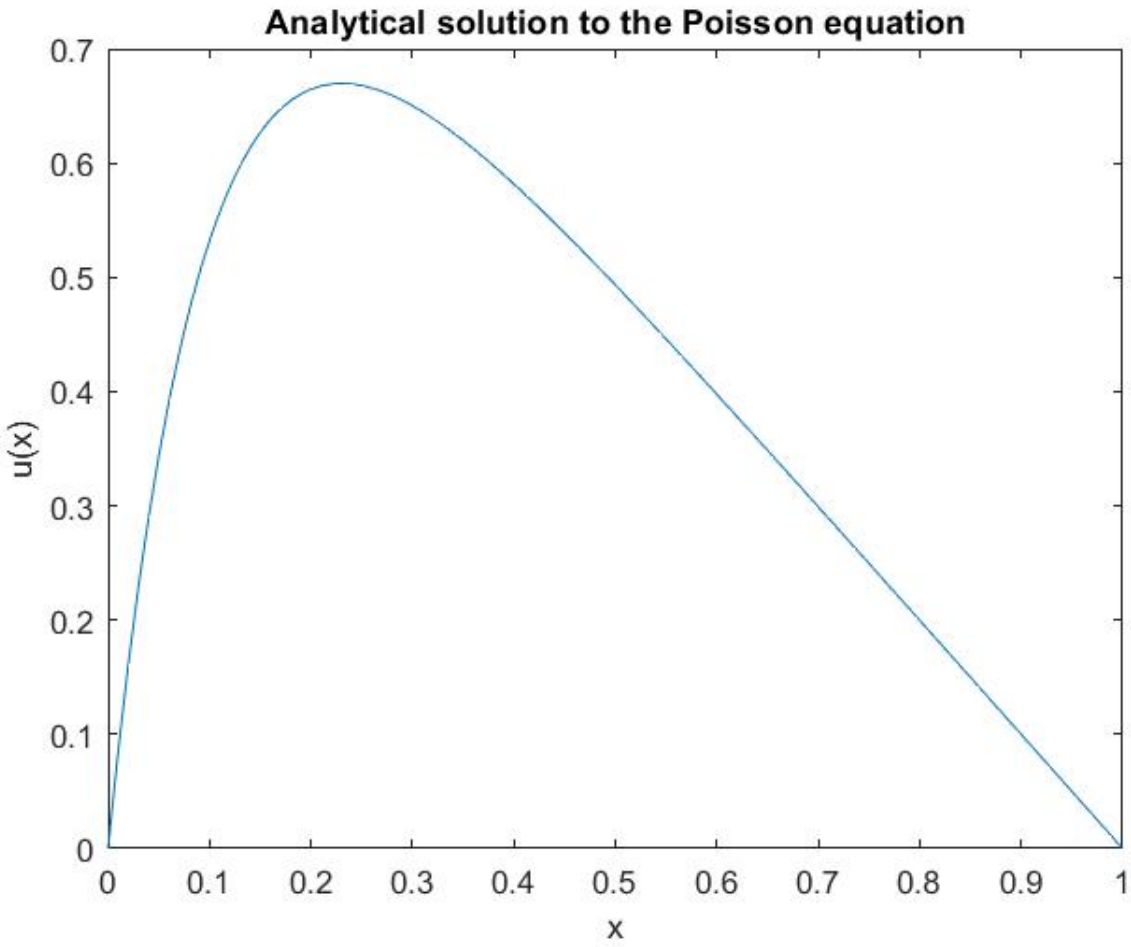

Figure 1: Plot of the solution to the Poisson equation

The file can be found in the Github depository for this project and with the tables mentioned earlier included when implementing the code in MATLAB.

4 Problem 3

The question posed in the problem asks us to derive a discretized version of the Poisson equation. A discrete form of the second derivative is:

$$u''(x) = \frac{u(x+h) - 2 * u(x) + u(x-h)}{h^2} \tag{5}$$

The discrete version of the Poisson equation can be found readily on the web or in textbooks for discrete calculus. By replacing the operator u" in equation (5) with the second difference approximation, one gets:

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f(x_i) \tag{6}$$

for 1 < i < n.The endpoints are fixed to zero. Meaning that we start at 1 rather than 0, and end at n rather than n + 1, because those are 0.

5 Problem 4

This problem wants us to show that we can rewrite the discretized equation as a matrix equation on the form:

$$A\overrightarrow{v} = \overrightarrow{g} \tag{7}$$

We write out the terms and move the h^2 on to the left side of the equation.

$$f(1) = -u_0 + 2u_1 - u_2 \tag{8}$$

$$f(2) = -u_1 + 2u_2 - u_3 \tag{9}$$

$$f(3) = -u_2 + 2u_3 - u_4 \tag{10}$$

$$f(n) = -u_{n-1} + 2u_n - u_{n+1} \tag{11}$$

In matrix form this becomes: $\begin{bmatrix} 2 & -1 & 0 & .... & 0 \\ -1 & 2 & -1 & .... & 0 \\ 0 & -1 & 2 & .... & 0 \end{bmatrix}$ We were also supposed to explain how an element of $\overrightarrow{g}$ is related to the variables in the differential equation. This comes from linear algebra where systems of linear equations, like those for us, are solved by the use of matrices.

6 Problem 5

6.1

n is the grid points and corresponds to the h in the equation (5), while the vector length is the number of elements in the vector. If they are grid vectors, then the matrix must have length(m) rows and length(m) columns. Because both variables have the same length, we get a n x n matrix. And in this assignment, m = n.

6.2

We find the interior solution. And not the end points.

7 Problem 6

7.1

The question posed asks us to write down an algorithm for solving $A\overrightarrow{v} = \overrightarrow{g}$. Previously, I've used the notation of u instead of the required v, and will continue to do so here. The general algorithm for solving Au = f needs a forward substitution and a backward substitution.

We take a 4 x 4 matrix given as: $\begin{bmatrix} b_1 & c_1 & 0 & 0 & f_1 \\ a_1 & b_2 & c_2 & 0 & f_2 \\ 0 & a_2 & b_3 & c_3 & f_3 \\ 0 & 0 & a_3 & b_4 & f_4 \end{bmatrix}$

The forward substitution gives us a general $\hat{b_i}$ and $\hat{f_i}$.

$$\hat{b_i} = b_i - \frac{c_{i-1}a_{i-1}}{\hat{b_{i-1}}} \tag{12}$$

$$\hat{f_i} = f_i - \frac{\hat{f_{i-1}}a_{i-1}}{\hat{b_{i-1}}} \tag{13}$$

A backward substitution gives us:

$$u_i = \frac{\hat{f_i} - c_i u_{i+1}}{\hat{b_i}} \tag{14}$$

A for-loop for i = 2, 3....n, and i = n-1, n-2.... is done on each substitution respectively.

7.2

The second part of this problem wants us to find the number of floating-operations (FLOPs) for the algorithm. 5 FLOPs are done in (12) and (13), because there are 5 i subscripts, and 3 FLOPs are done in (14). Each loop is run n - 1 times, making the (8n - 7) FLOPs. For large n, the constant is ignored and the result is 8n FLOPs.

8 Problem 7

8.1

This particular problem, a), is divided in two. We are to write a program that uses the general algorithm to solve $A\overrightarrow{u} = \overrightarrow{g}$, and to write the solution to a file. I wrote a general algorithm for the first part. This is given on the Github link provided on the last page of this text. The second part of the question was to write a function for writing the results to a text file. I sadly do not have a plot of the different n-values due to the lack of a proper installed compiler on my computer. I joined the course only in the last possible day (first of September) and have not been able to get a working compiler. I'm also forced to stay in during the Covid-19 pandemic due to being diabetic. So, I only have one n variable besides the analytical value. The plot of that can be seen on figure 2. The results obtained during the code used were:

f = (0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0)

f = (0, 0, 0, 0.00614025, 0.145331, 0.99828, 1.7101, 0.985075, 0.143801, 0.0104494, 0.000759311, 0, 0)

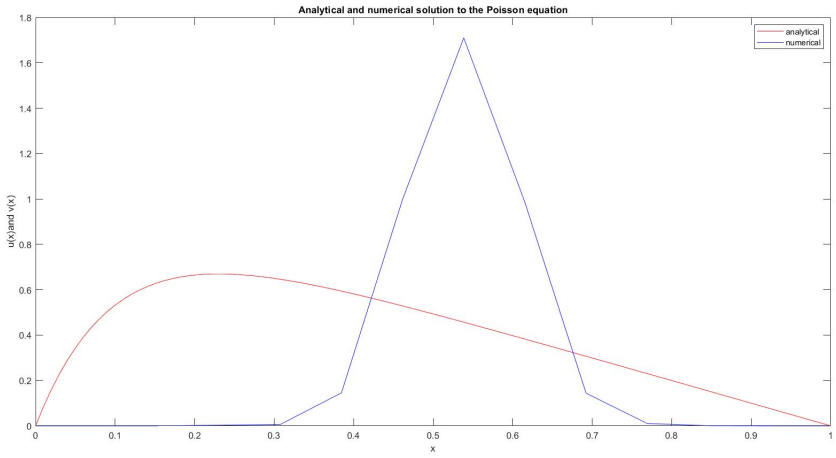

Figure 2: Shows my attempt at both numerical and analytical solution of the Poisson equation

9 Problem 8

Obviously I do not have a good code for the relationship between the error for the approximation and analytical result. But I would expect that because the x value is defined between 0 and 1, that the error is not as large as if it was defined to a value above 1.

10 Problem 9

10.1

The problem wants us to specialize the algorithm from Problem 6 to the special case where A is specified by the signature (-1, 2, -1), with $\overrightarrow{a}$ = (-1, -1, -1), $\overrightarrow{b}$ = (2, 2, 2) and $\overrightarrow{c}$ = (-1, -1, -1). We insert these values into the equations (12), (13) and (14), and get the following expressions.

$$\hat{b_i} = 2 - \frac{1}{\hat{b_i} - 1} \tag{15}$$

$$\hat{f}_i = f_i + \frac{\hat{f_{i-1}}}{b_{i-1}} \tag{16}$$

We therefore get:

$$\hat{f}_i = f_i + \frac{\hat{f_{i-1}}}{\hat{b_{i-1}}} \tag{17}$$

$$u_i = \frac{\hat{f}_i + u_{i+1}}{\hat{b_i}} \tag{18}$$

These are the specialized algorithms where $\hat{b_i}$ is to be recurrence relations.

10.2

The loops use 2 FLOPs each time they run, and because they run n - 1 times, we get 4n - 3 FLOPs. For large, we get 4n FLOPs.

11 Problem 10

I could not get the code to work properly due to lack of a functioning compiler, but based on the FLOPs, we get that the timing test will show that the specialized algorithm will take considerably shorter time. For larger n we have 8n FLOPs for the general algorithm, and 4n FLOPs for the specialized algorithm.

12 Problem 11

I believe it will take longer time to use a LU-decomposition that has to use the entire matrix compared to the specialized algorithm that stores only three diagonal arrays.

13 Github

The work done in this project can be found at this url:

https://github.com/FurkanYekmal/FYS3150