FYS3150 - Project 1

Frida Larsen

I. INTRODUCTION

Reference code in github repository.

The one-dimensional Poisson equation is given by

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

with Dirichlet boundary conditions

$$u(0) = u(1) = 0 \tag{2}$$

on the interval $x \in (0, 1)$.

II. THEORY

The second derivative of a general discretized function g_i can be approximated by

$$g_i'' \approx \frac{g_{i+1} + g_{i-1} - 2g_i}{h^2} \tag{3}$$

The discretized version of the Poisson equation (1) then becomes

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} = f_i, \tag{4}$$

where we have defined the discretized approximation to u and x as u_i and x_i respectively, such that $x_i = ih$, $x_0 = 0$ and $x_{n+1} = 1$. We also have $f_i = f(x_i)$. The step length h is defined as

$$h = \frac{1}{n+1}, \tag{5}$$

where n is the total number of grid points on the interval (0,1). The Dirichlet boundary conditions (2) become

$$u_0 = u_{n+1} = 0. \tag{6}$$

The discrete Poisson equation can be further simplified as

$$-u_{i+1} - ui - 1 + 2u_i = h^2 f_i.$$

By introducing $\tilde{b}_i = h^2 f_i$ and remembering that $u_0 = 0$ the Poisson equation for the first few i-values are

$$\begin{aligned}
i = 1:& \quad 2u_1 - u_2 = \tilde{b}_1 \\
i = 2:& \quad -u_1 + 2u_2 - u_3 = \tilde{b}_2 \\
i = 3:& \quad -u_2 + 2u_3 - u_4 = \tilde{b}_3 \\
&\dots
\end{aligned}$$

This set of equations can be rewritten as a matrix equation

$$\mathbf{A}\mathbf{u} = \tilde{\mathbf{b}} \tag{7}$$

with

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{pmatrix}. \tag{8}$$

III. METHOD

Include closed-form solution here as a part of unit test maybe??

In order to develop an algorithm for solving the matrix equation (7) we will begin with a general $n \times n$ tridiagonal matrix

$$\tilde{\mathbf{A}} = \begin{pmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots \\ a_1 & b_2 & c_2 & 0 & \dots & \dots \\ 0 & a_2 & b_3 & c_3 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & 0 & a_{n-2} & b_{n-1} & c_{n-1} \\ \dots & \dots & \dots & 0 & a_{n-1} & b_n \end{pmatrix} \tag{9}$$

Reference Tridiagonal Matrix Algorithm on Wikipedia.

Algorithm 1: The first step is a forward substitution,

$$c_i' = \begin{cases} \frac{c_i}{b_i} & ; \quad i = 1 \\ \frac{c_i}{b_i - a_i c_{i-1}'} & ; \quad i = 2, 3, \dots, n-1 \end{cases} \tag{10}$$

and

$$\tilde{b}_i' = \begin{cases} \frac{\tilde{b}_i}{b_i} & ; \quad i = 1 \\ \frac{\tilde{b}_i - a_i \tilde{b}_{i-1}'}{b_i - a_i c_{i-1}'} & ; \quad i = 2, 3, \dots, n \end{cases} \tag{11}$$

The second step is to obtain the solution by back substitution,

$$u_n = \tilde{b}_n'; \quad i = n \tag{12}$$

$$u_i = \tilde{b}_i' - c_i' u_{i+1}; \quad i = n-1, n-2, \dots, 1. \tag{13}$$

Algorithm 2: Although the above algorithm works for the matrix belonging to our Poisson equation, the generalized $\tilde{\mathbf{A}}$ is actually more complicated than our original

A. A more accurate generalization would be

$$\mathbf{A}_g = \begin{pmatrix} b_1 & a_1 & 0 & 0 & \dots & 0 \\ a_2 & b_2 & a_2 & 0 & \dots & 0 \\ 0 & a_3 & b_3 & a_3 & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & a_{n-1} \\ \dots & \dots & \dots & \dots & a_n & b_n \end{pmatrix}, \tag{14}$$

where the upper and lower diagonals consist of the same elements. We observe that

$$\mathbf{A} \underset{\sim}{II - \frac{a_2}{b_2} I} \begin{pmatrix} b_1 & a_1 & 0 & 0 & \dots & 0 \\ 0 & \tilde{b}_2 & a_2 & 0 & \dots & 0 \\ 0 & a_3 & b_3 & a_3 & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & a_{n-1} \\ \dots & \dots & \dots & \dots & a_n & b_n \end{pmatrix},$$

with $\tilde{b}_2 = b_2 - \frac{a_2 a_1}{b_1}$. Repeating this process for all rows 1 to n results in the lower diagonal being replaced by 0's only and the diagonal b's being replaced by

$$\tilde{b}_i = \begin{cases} b_1, & i = 1 \\ b_i - \frac{a_i a_{i-1}}{\tilde{b}_{i-1}}, & i = 2, 3, \dots, n \end{cases} \tag{15}$$

Applying this process to the right hand side of equation 7 we get

$$\tilde{f}_i = \begin{cases} f_1, & i = 1 \\ f_i - \frac{a_i \tilde{f}_{i-1}}{\tilde{b}_{i-1}}, & i = 2, 3, \dots, n \end{cases} \tag{16}$$

Substituting these expressions backwards into the matrix equation 7 we find the solution

$$u_i = \begin{cases} \frac{\tilde{f}_i}{\tilde{b}_j}, & i = n \\ \frac{\tilde{f}_i - a_i u_{i+1}}{\tilde{b}_i}, & i = n-1, n-2, \dots, 1 \end{cases} \tag{17}$$

Although these expressions are much simpler than those for algorithm 1, there are further simplifications to be made. For our original matrix (8) all the a's are -1 and all the b's are 2. Inserting this yields

$$\begin{aligned} \tilde{b}_i &= \begin{cases} 2, & i = 1 \\ 2 - \frac{1}{\tilde{b}_{i-1}}, & i = 1, 2, \dots, n \end{cases} \\ &= \frac{i+1}{i}, \qquad i = 1, 2, \dots, n, \end{aligned} \tag{18}$$

which leads to

$$\begin{aligned} \tilde{f}_i &= \begin{cases} f_1, & i = 1 \\ f_i + \frac{\tilde{f}_{i-1}}{i/(i-1)}, & i = 2, 3, \dots, n \end{cases} \\ &= f_i + \frac{(i-1)\tilde{f}_{i-1}}{i}, \quad i = 1, 2, \dots, n \end{aligned} \tag{19}$$

and finally

$$u_{i-1} = \frac{i-1}{i}(\tilde{f}_{i-1} + u_i), \ i = n, \dots, 2 \tag{20}$$

with $u_n = \tilde{f}_n / \tilde{b}_n$.

In order to test these algorithms, we will use a function for the right hand side of the matrix equation (7) given by

$$\tilde{b}(x) = h^2 100 e^{-10x}, \tag{21}$$

with closed-form solution

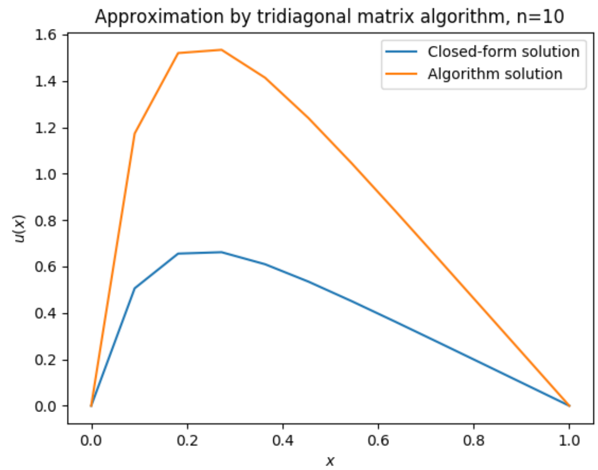
$$u_s(x) = 1 - (1 - e^{-10})x - e^{-10x}. \tag{22}$$

IV. RESULTS

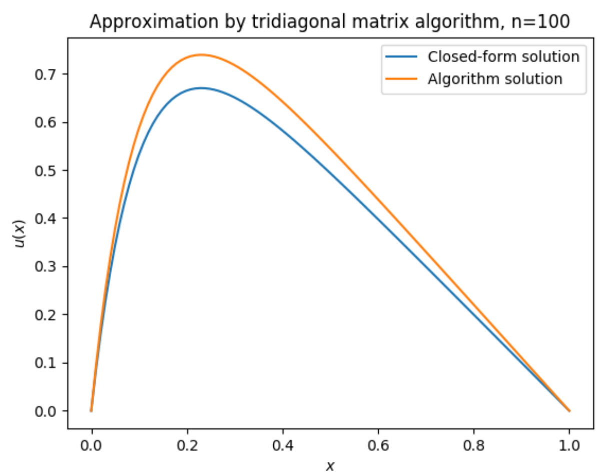
Figure 1 shows plots of the algorithm solution as compared to the closed-form solution of the matrix equation (7) with the general $\tilde{\mathbf{A}}$ as given by equation 9 for three different sized matrices. We see that the curves with more grid points are closer to the closed-form solution.

V. DISCUSSION

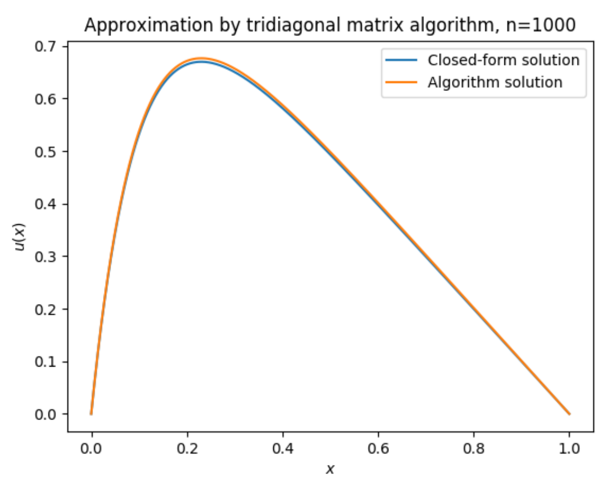
VI. CONCLUSION



(a) Comparison using n=10 grid points.



(b) Comparison using n=100 grid points.



(c) Comparison using n=1000 grid points.

FIG. 1. Comparison of the tridiagonal matrix algorithm and the analytical solution for the function $\tilde{b}$ (21) for different sized matrices.