Project 1

Alessio Canclini, Filip von der Lippe

(Dated: September 9, 2022)

List a link to your github repository here!

PROBLEM 1

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

- source term: $f(x) = 100e^{-10x}$
- x range $x \in [0, 1]$
- boundary conditions: $u(0) = 0$ and $u(1) = 0$

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

Checking analytically that an exact solution to Eq. 1 is given by Eq. 2.

$$\begin{aligned}
\frac{du}{dx} &= 1 - e^{-10} + 10e^{-10x} \\
\frac{d^2u}{dx^2} &= -100e^{-10x} \\
-\frac{d^2u}{dx^2} &= 100e^{-10x} \\
-\frac{d^2u}{dx^2} &= f(x)
\end{aligned}$$

PROBLEM 2

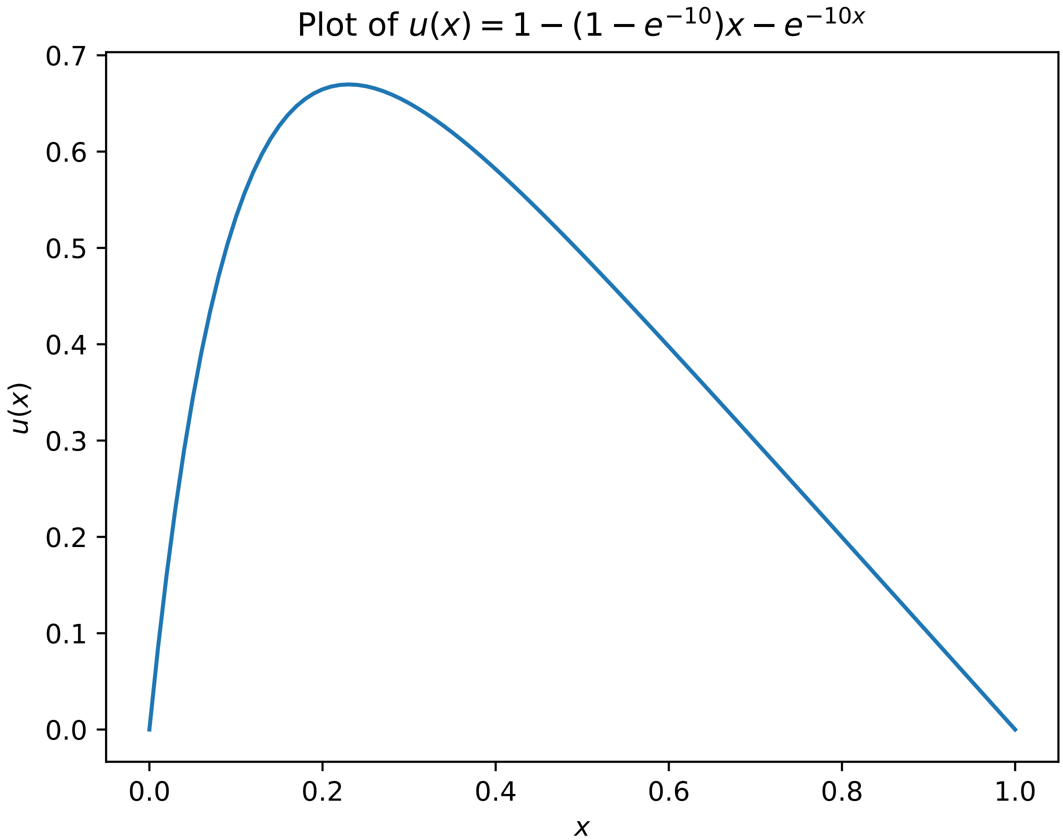

FIG. 1. Plot of $u(x)$.

PROBLEM 3

By using the taylor approximation of the second derivative we can discretize the second derivative in the Poission equation:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + O(h^2)$$

Here we have the stepsize h and the truncation error O. The one-dimensional Poisson equation can then be written for the approximated version of u as v like:

$$-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} = f_i \tag{3}$$

PROBLEM 4

We can rewrite the discretized equation as a matrix equation for $n+1$ number of points and $n-1$ unknown points (v_0 and v_n are known) with the $n-1 \times n-1$ matrix $\boldsymbol{A}$. We rewrite the discretized Poisson funcion:

$$\begin{aligned} 2v_1 - v_2 &= f_1 h^2 \\ -v_1 + 2v_2 - v_3 &= f_2 h^2 \\ &\vdots \\ -v_{n-3} + 2v_{n-2} - v_{n-1} &= f_{n-2} h^2 \\ -v_{n-2} + 2v_{n-1} &= f_{n-1} h^2 \end{aligned}$$

This can be written in terms of the following matrix equation where we rewrite $f_i h^2$ for $i = 1, 2, \ldots n-1$ as g_i

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \ldots & 0 \\ -1 & 2 & -1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & -1 & 2 & -1 \\ 0 & 0 & \ldots & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{n-2} \\ g_{n-1} \end{bmatrix}$$

PROBLEM 5

a) Since the vector $\vec{v*}$ of length m represents a complete solution of the descretized Poisson equation it contains all values in $\vec{v}$ in addition to the boundary conditions. The relation bethween n and m is therfore $m = n + 2$.

b) Solving $\boldsymbol{A}\vec{v} = \vec{g}$ for $\vec{v}$ gives us all but the first an last value in $\vec{v*}$. So all but the boundary values.

PROBLEM 6

a) We can use the Thomas algorithm to row reduce the matrix $\boldsymbol{A}$ to give us a solution of the equation $\boldsymbol{A}\vec{v} = \vec{g}$. This is done in two steps. We first define the three diagonals as three vectors. The sub diagonal $\vec{a}$ the diagonal $\vec{b}$ and the superdiagonal $\vec{c}$ where the i element in theese vectors corresponds to the i row of the matrix $\boldsymbol{A}$. We define n unknowns and the matrix $\boldsymbol{A}$ as $n \times n$

i) The first step is forwards substitution. We define a numer w for each step and overwrite the i element of both b and g starting at index 2. This means overwrite the values in the original vectors b and g, but at the same time we dont have to define new vectors to store new values. For large n this will reduce both the computation time and memory usage for the data machine

for $i = 2, .., n$:

$$\begin{aligned} w &= \frac{a_i}{b_{i-1}} \\ b_i &= b_i - w c_{i-1} \\ g_i &= g_i - w g_{i-1} \end{aligned}$$

ii) The second and last step is back substitution where we find an expression for $\vec{v}$. We start at our last element and work our way backwards:

$$\begin{aligned} v_n &= \frac{g_n}{b_n} \\ v_i &= \frac{g_i - c_i v_{i+1}}{b_i} \quad \text{for } i = n-1, ..., 1 \end{aligned}$$

b) We find the number of FLOPs for this algorithm by counting the number of floating point operations the computer has to do. For the first step we have 3 FLOPs (1 subtraction 1 division and one multiplication) for defining b_i and g_i each. Since we loop this operation $n-1$ times we end up with a totalt number of $6(n-1)$ FLOPs.

For back substitution we have 1 FLOP calculating v_n, and three FLOPs calculating v_i $n-1$ times. This gives us $3(n-1)+1$ FLOPs

The total FLOPs of the general algorithm is $9n-8$

PROBLEM 7

PROBLEM 8

PROBLEM 9

a) For the special case we dont need to do new computations for every i element of the vectors $\vec{a}$ and $\vec{c}$ and thus don't need to assign and use these variabeles.

i) The first step is forwards substitution. We have to define $b_1 = 2$ and then loop over the rest:

$$b_i = b_i + \frac{1}{b_{i-1}} \quad \text{for } i = 2, .., n$$
$$g_i = g_i + \frac{1}{b_{i-1}} g_{i-1} \quad \text{for } i = 2, .., n$$

ii) The second and last step is back substitution where we find an expression for $\vec{v}$. We start at our last element and work our way backwards:

$$v_n = \frac{g_n}{b_n}$$
$$v_i = \frac{g_i + v_{i+1}}{b_i} \quad \text{for } i = n-1, ..., 1$$

b) Since we know that $a, c = -1$ we are abel to reduce the amount of FLOPs compared to the general algorithm. For the forward substitution we have $2(n-1)$ FLOPs to compute b_i and the same for g_i. The back substitution requires 1 FLOP for v_n and $2(n-1)$ FLOPs for v_i. This gives us a total of $6n-5$ FLOPs for the special algorithm.

PROBLEM 10

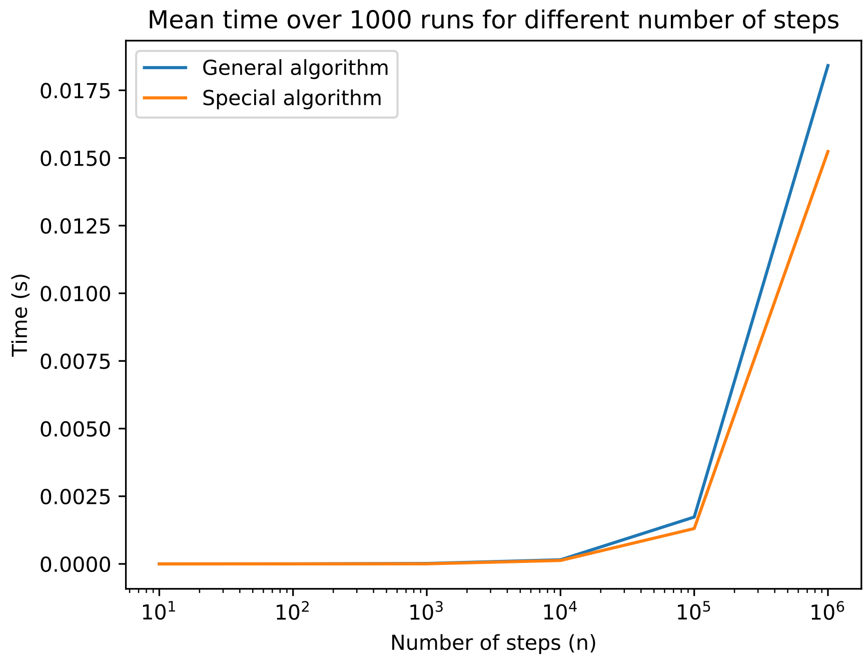

FIG. 2.

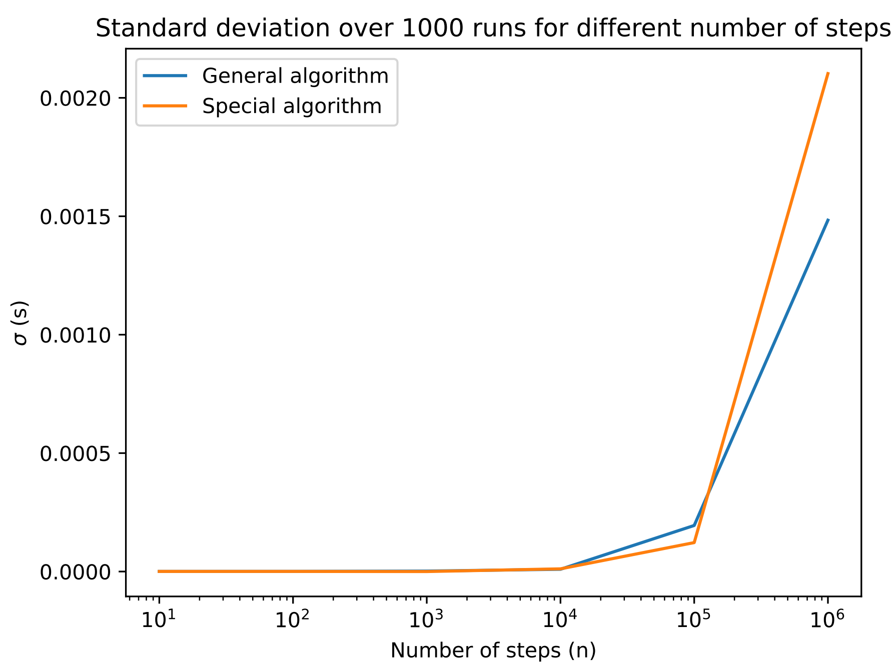

FIG. 3.