Project 1

Cory Alexander Balaton & Janita Ovidie Sandtrøen Willumsen
(Dated: September 12, 2023)

https://github.uio.no/FYS3150-G2-2023/Project-1

PROBLEM 1

First, we rearrange the equation.

$$\begin{aligned}
-\frac{d^2u}{dx^2} &= 100e^{-10x} \\
\frac{d^2u}{dx^2} &= -100e^{-10x}
\end{aligned}$$

Now we find $u(x)$.

$$\begin{aligned}
u(x) &= \int\int \frac{d^2u}{dx^2} dx^2 \\
&= \int\int -100e^{-10x} dx^2 \\
&= \int \frac{-100e^{-10x}}{-10} + c_1 dx \\
&= \int 10e^{-10x} + c_1 dx \\
&= \frac{10e^{-10x}}{-10} + c_1 x + c_2 \\
&= -e^{-10x} + c_1 x + c_2
\end{aligned}$$

Using the boundary conditions, we can find c_1 and c_2

$$\begin{aligned}
u(0) &= 0 \\
-e^{-10\cdot 0} + c_1 \cdot 0 + c_2 &= 0 \\
-1 + c_2 &= 0 \\
c_2 &= 1
\end{aligned}$$

$$\begin{aligned}
u(1) &= 0 \\
-e^{-10\cdot 1} + c_1 \cdot 1 + c_2 &= 0 \\
-e^{-10} + c_1 + c_2 &= 0 \\
c_1 &= e^{-10} - c_2 \\
c_1 &= e^{-10} - 1
\end{aligned}$$

Using the values that we found for c_1 and c_2, we get

$$\begin{aligned}
u(x) &= -e^{-10x} + (e^{-10} - 1)x + 1 \\
&= 1 - (1 - e^{-10})x - e^{-10x}
\end{aligned}$$

PROBLEM 2

The code for generating the points and plotting them can be found under.
Point generator code: https://github.uio.no/FYS3150-G2-203/Project-1/blob/main/src/analyticPlot.cpp
Plotting code: https://github.uio.no/FYS3150-G2-2023/Project-1/blob/main/src/analyticPlot.py

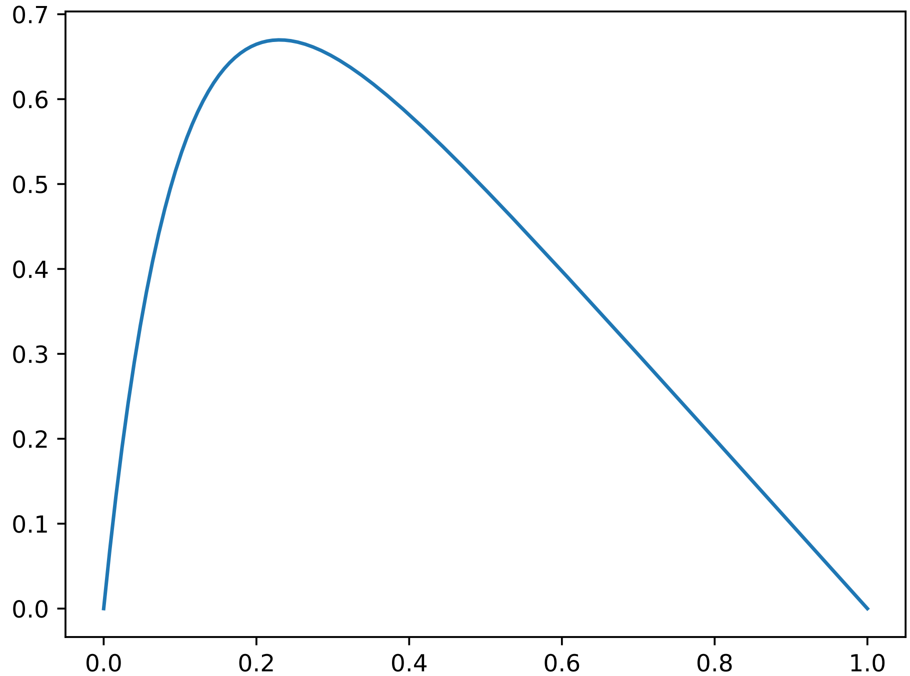

FIG. 1. Plot of the analytical solution $u(x)$.

PROBLEM 3

To derive the discretized version of the Poisson equation, we first need the Taylor expansion for $u(x)$ around x for $x+h$ and $x-h$.

$$u(x+h) = u(x) + u'(x)h + \frac{1}{2}u''(x)h^2 + \frac{1}{6}u'''(x)h^3 + \mathcal{O}(h^4)$$

$$u(x-h) = u(x) - u'(x)h + \frac{1}{2}u''(x)h^2 - \frac{1}{6}u'''(x)h^3 + \mathcal{O}(h^4)$$

If we add the equations above, we get this new equation:

$$\begin{aligned}
u(x+h) + u(x-h) &= 2u(x) + u''(x)h^2 + \mathcal{O}(h^4) \\
u(x+h) - 2u(x) + u(x-h) + \mathcal{O}(h^4) &= u''(x)h^2 \\
u''(x) &= \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + \mathcal{O}(h^2) \\
u_i''(x) &= \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \mathcal{O}(h^2)
\end{aligned}$$

We can then replace $\frac{d^2u}{dx^2}$ with the RHS (right-hand side) of the equation:

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f(x) \\ \frac{-u_{i+1} + 2u_i - u_{i-1}}{h^2} + \mathcal{O}(h^2) &= f_i \end{aligned}$$

And lastly, we leave out $\mathcal{O}(h^2)$ and change u_i to v_i to differentiate between the exact solution and the approximate solution, and get the discretized version of the equation:

$$\frac{-v_{i+1} + 2v_i - v_{i-1}}{h^2} = 100e^{-10x_i}$$

PROBLEM 4

The value of $u(x_0)$ and $u(x_1)$ is known, using the discretized equation we solve for $\vec{v}$. This will result in a set of equations

$$\begin{aligned} -v_0 + 2v_1 - v_2 &= h^2 \cdot f_1 \\ -v_1 + 2v_2 - v_3 &= h^2 \cdot f_2 \\ &\vdots \\ -v_{m-2} + 2v_{m-1} - v_m &= h^2 \cdot f_{m-1} \end{aligned}$$

where $v_i = v(x_i)$ and $f_i = f(x_i)$. Rearranging the first and last equation, moving terms of known boundary values to the RHS

$$\begin{aligned} 2v_1 - v_2 &= h^2 \cdot f_1 + v_0 \\ -v_1 + 2v_2 - v_3 &= h^2 \cdot f_2 \\ &\vdots \\ -v_{m-2} + 2v_{m-1} &= h^2 \cdot f_{m-1} + v_m \end{aligned}$$

We now have a number of linear eqations, corresponding to the number of unknown values, which can be represented as an augmented matrix

$$\left[\begin{array}{cccccc|c} 2v_1 & -v_2 & 0 & \dots & & 0 & g_1 \\ -v_1 & 2v_2 & -v_3 & 0 & & & g_2 \\ 0 & -v_2 & 2v_3 & -v_4 & & & g_2 \\ \vdots & & & \ddots & & \vdots & \vdots \\ 0 & & & -v_{m-2} & & 2v_{m-1} & g_{m-1} \end{array}\right]$$

Since the boundary values are equal to 0 the RHS can be renamed $g_i = h^2 f_i$ for all i. An augmented matrix can be represented as $\boldsymbol{A}\vec{x} = \vec{b}$. In this case $\boldsymbol{A}$ is the coefficient matrix with a tridiagonal signature $(-1, 2, -1)$ and dimension $n \times n$, where $n = m - 2$.

PROBLEM 5

a & b)

$n = m - 2$ since when solving for $\vec{v}$, we are finding the solutions for all the points that are in between the boundaries and not the boundaries themselves. $\vec{v}^*$ on the other hand includes the boundary points.

PROBLEM 6

a)

Renaming the sub-, main-, and supdiagonal of matrix $\boldsymbol{A}$

$$\vec{a} = [a_2, a_3, ..., a_{n-1}, a_n]$$
$$\vec{b} = [b_1, b_2, b_3, ..., b_{n-1}, b_n]$$
$$\vec{c} = [c_1, c_2, c_3, ..., c_{n-1}]$$

Following Thomas algorithm for gaussian elimination, we first perform a forward sweep followed by a backward sweep to obtain $\vec{v}$

Algorithm 1 General algorithm

```
procedure FORWARD SWEEP(a⃗, b⃗, c⃗)
    n ← length of b⃗
    b̂⃗, ĝ⃗ ← vectors of length n.
    b̂_1 ← b_1                      ▷ Handle first element in main diagonal outside loop
    ĝ_1 ← g_1
    for i = 2, 3, ..., n do
        d ← a_i / b̂_{i-1}          ▷ Calculating common expression
        b̂_i ← b_i − d · c_{i-1}
        ĝ_i ← g_i − d · ĝ_{i-1}
    return b̂⃗, ĝ⃗
procedure BACKWARD SWEEP(b̂⃗, ĝ⃗)
    n ← length of b̂⃗
    v⃗ ← vector of length n.
    v_n ← ĝ_n / b̂_n
    for i = n − 1, n − 2, ..., 1 do
        v_i ← (ĝ_i − c_i · v_{i+1}) / b̂_i
    return v⃗
```

b)

Counting the number of FLOPs for the general algorithm by looking at one procedure at a time. For every iteration of i in forward sweep we have 1 division, 2 multiplications, and 2 subtractions, resulting in $5(n-1)$ FLOPs. For backward sweep we have 1 division, and for every iteration of i we have 1 subtraction, 1 multiplication, and 1 division, resulting in $3(n-1)+1$ FLOPs. Total FLOPs for the general algorithm is $8(n-1)+1$.

PROBLEM 7

a)

The code can be found at https://github.uio.no/FYS3150-G2-2023/Project-1/blob/coryab/final-run/src/generalAlgorithm.cpp

b)

Increasing the number of steps results in an approximation close to the exact solution.

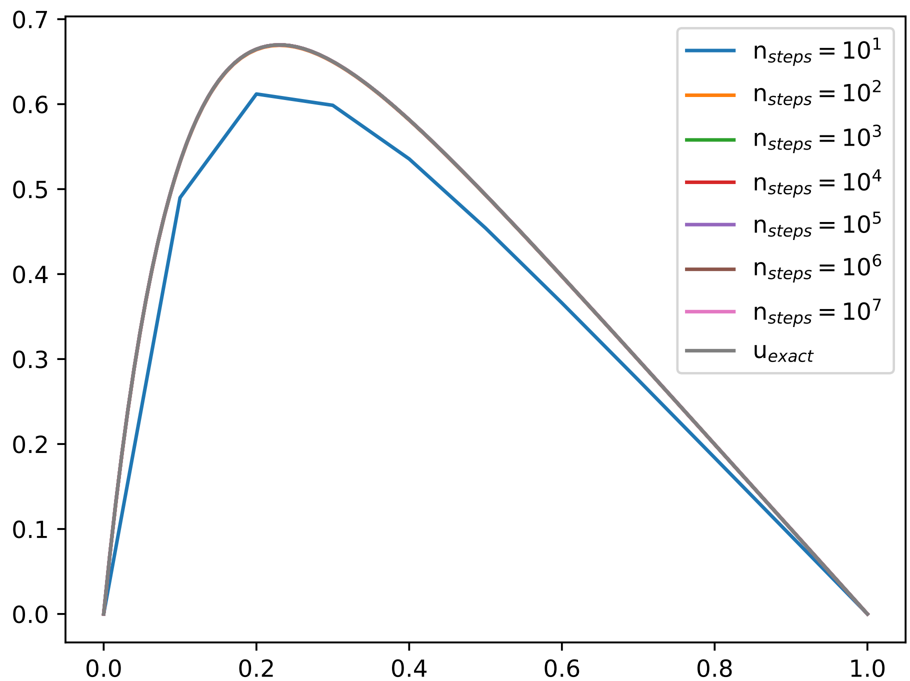

FIG. 2. Plot showing the numeric solution of u_{approx} for n_{steps} and the exact solution u_{exact}.

PROBLEM 8

a)

Increasing number of steps result in a decrease of absolute error.

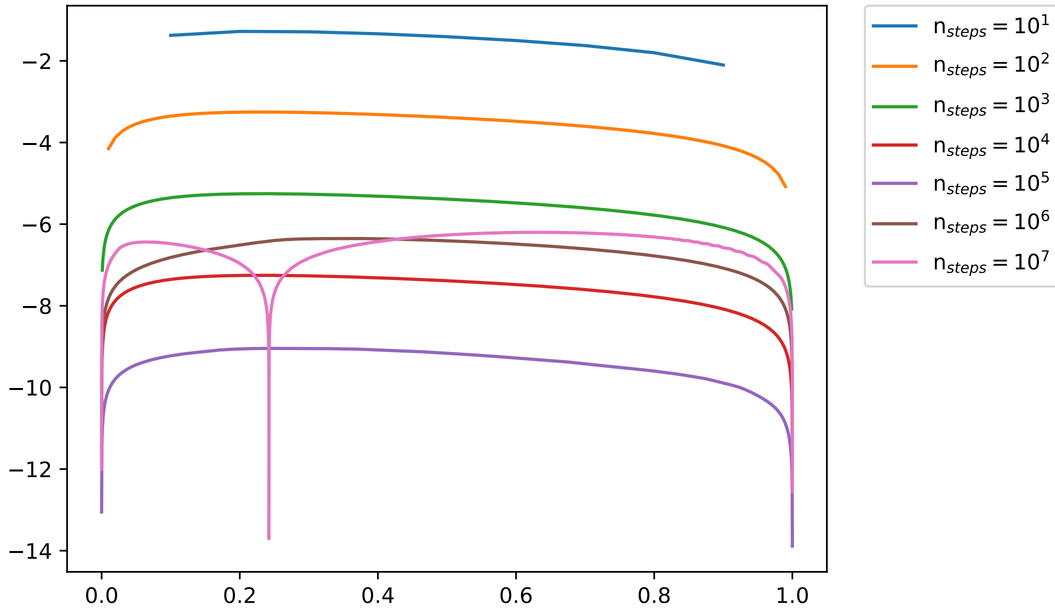

FIG. 3. Absolute error for different step sizes n_{steps}.

b)

Increasing number of steps also result in a decrease of absolute error.

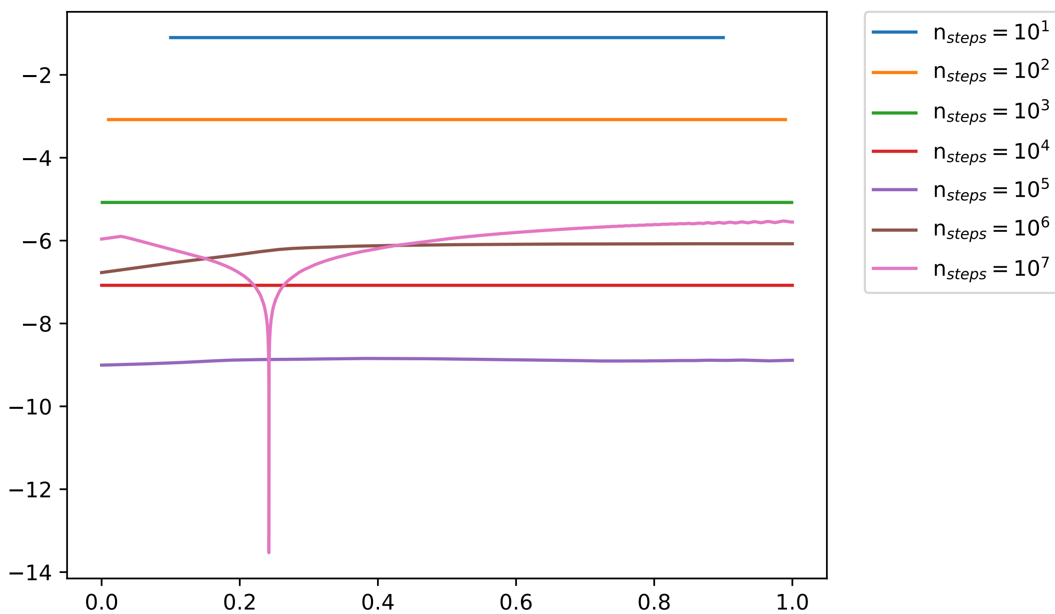

FIG. 4. Relative error for different step sizes n_{steps}.

c)

Increasing number of steps result in a decrease of maximum relative error, up to a certain number of steps. At $n_{steps} \approx 10^5$ the maximumrelative error increase. This can be related to loss of numerical precicion when step size is small.

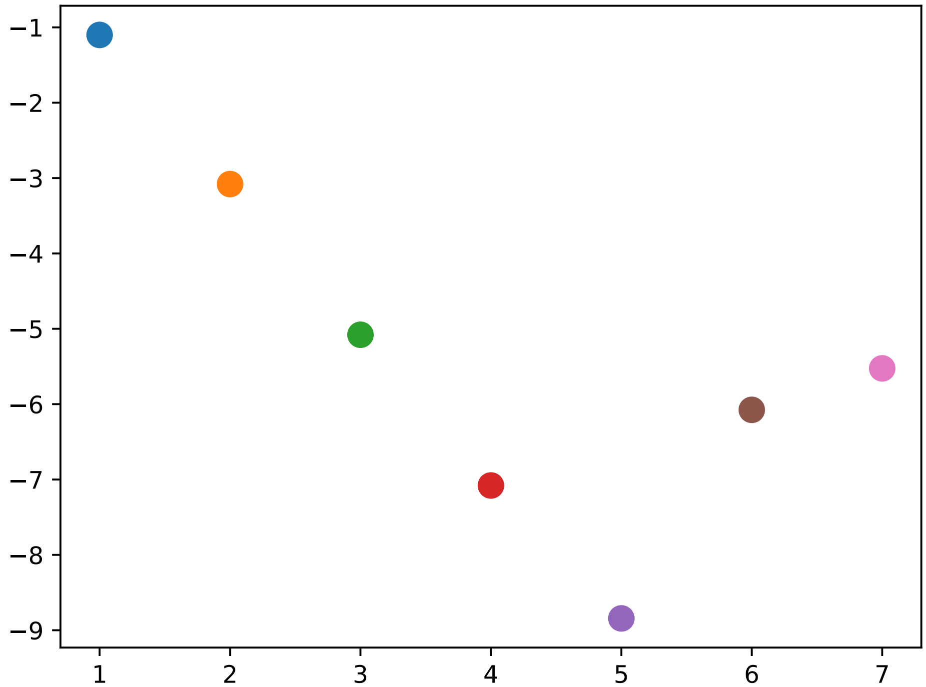

FIG. 5. Maximum relative error for each step sizes n_{steps}.

PROBLEM 9

a)

The special algorithm does not require the values of all a_i, b_i, c_i. We find the values of $\hat{b}_i$ from simplifying the general case

$$\hat{b}_i = b_i - \frac{a_i \cdot c_{i-1}}{\hat{b}_{i-1}}$$
$$\hat{b}_i = 2 - \frac{1}{\hat{b}_{i-1}}$$

Calculating the first values to see a pattern

$$\hat{b}_1 = 2$$
$$\hat{b}_2 = 2 - \frac{1}{2} = \frac{3}{2}$$
$$\hat{b}_3 = 2 - \frac{1}{\frac{3}{2}} = \frac{4}{3}$$
$$\hat{b}_4 = 2 - \frac{1}{\frac{4}{3}} = \frac{5}{4}$$
$$\vdots$$
$$\hat{b}_i = \frac{i+1}{i} \qquad \text{for } i = 1, 2, ..., n$$

Algorithm 2 Special algorithm

procedure FORWARD SWEEP($\vec{b}$)
  $n \leftarrow$ length of $\vec{b}$
  $\vec{\hat{b}}$, $\vec{\hat{g}} \leftarrow$ vectors of length n.
  $\hat{b}_1 \leftarrow 2$ ▷ Handle first element in main diagonal outside loop
  $\hat{g}_1 \leftarrow g_1$
  for $i = 2, 3, ..., n$ **do**
    $\hat{b}_i \leftarrow \frac{i+1}{i}$
    $\hat{g}_i \leftarrow g_i + \frac{\hat{g}_{i-1}}{\hat{b}_{i-1}}$
  return $\vec{\hat{b}}$, $\vec{\hat{g}}$
procedure BACKWARD SWEEP($\vec{\hat{b}}$, $\vec{\hat{g}}$)
  $n \leftarrow$ length of $\vec{\hat{b}}$
  $\vec{v} \leftarrow$ vector of length n.
  $v_n \leftarrow \frac{\hat{g}_n}{\hat{b}_n}$
  for $i = n-1, n-2, ..., 1$ **do**
    $v_i \leftarrow \frac{\hat{g}_i + v_{i+1}}{\hat{b}_i}$
  return $\vec{v}$

b)

For every iteration of i in forward sweep we have 2 divisions, and 2 additions, resulting in $4(n-1)$ FLOPs. For backward sweep we have 1 division, and for every iteration of i we have 1 addition, and 1 division, resulting in $2(n-1)+1$ FLOPs. Total FLOPs for the special algorithm is $6(n-1)+1$.

PROBLEM 10

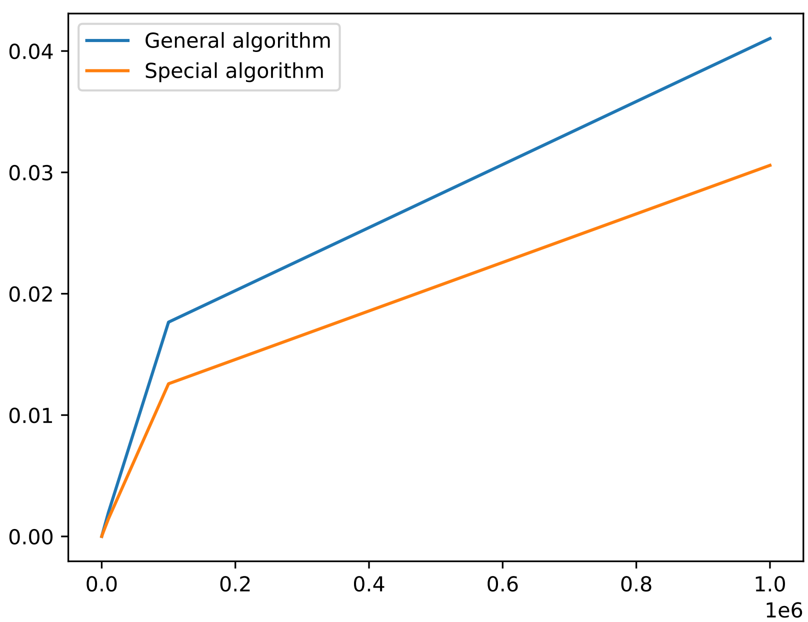

FIG. 6. Timing of general algorithm vs. special for step sizes n_{steps}