

FYS3150 - Project 1

Andrew Quan, Oskar Idland, Hishem Kløvnes, Håvard Skåli
(Dated: August 25, 2023)

[GitHub Repository](#)

PROBLEM 1

We have the one-dimensional Poisson equation

$$-\frac{d^2u}{dx^2} = 100e^{-10x}, \quad x \in [0, 1] \quad (1)$$

where the right hand side of the equation is the source term $f(x)$. We have the boundary conditions $u(0) = 0$ and $u(1) = 0$. Integrating both sides of (1) with respect to x twice we get

$$\begin{aligned} \frac{d^2u}{dx^2} &= -100e^{-10x} \\ \iint \left(\frac{d^2u}{dx^2} \right) dx^2 &= -100 \iint e^{-10x} dx^2 \\ \int \left(\frac{du}{dx} \right) dx &= -100 \int \left(-\frac{1}{10}e^{-10x} + C \right) dx \\ u(x) &= -100 \left(\frac{1}{100}e^{-10x} + Cx + D \right) \\ u(x) &= Cx + D - e^{-10x} \end{aligned} \quad (2)$$

Where C and D are some arbitrary constants. Invoking the boundary conditions we get

$$\begin{aligned} u(0) &= D - 1 = 0 \Rightarrow D = 1 \\ u(1) &= C + 1 - e^{-10} = 0 \Rightarrow C = e^{-10} - 1 \end{aligned}$$

Thus, plugging these constants into the general solution and rearranging the terms, the final solution becomes

$$u(x) = 1 - x(1 - e^{-10}) - e^{-10x} \quad (3)$$

just like we wanted to show.

PROBLEM 2

After creating a program that defines a vector of x -values between 1 and 100 and evaluates the exact solution $u(x)$ defined as in (3), we wrote these numbers with four decimals into a text file and plotted them against each other in Python. The resulting plot is shown in FIG. 1.

PROBLEM 3

Our aim is to derive a discretized version of the Poisson equation which lets us calculate approximate values v of the exact values u . To create this program we need to discretize the double derivative, which we can do by recalling its definition:

$$\frac{d^2u}{dx^2} = \lim_{dx \rightarrow 0} \frac{-u(x - dx) + 2u(x) - u(x + dx)}{dx^2} \quad (4)$$

If we let the infinitesimal value dx rather become a finite, small value Δx , we'll still be able to calculate the double derivative with good precision, as long as we don't let this value grow too large. Since this is an approximation, a

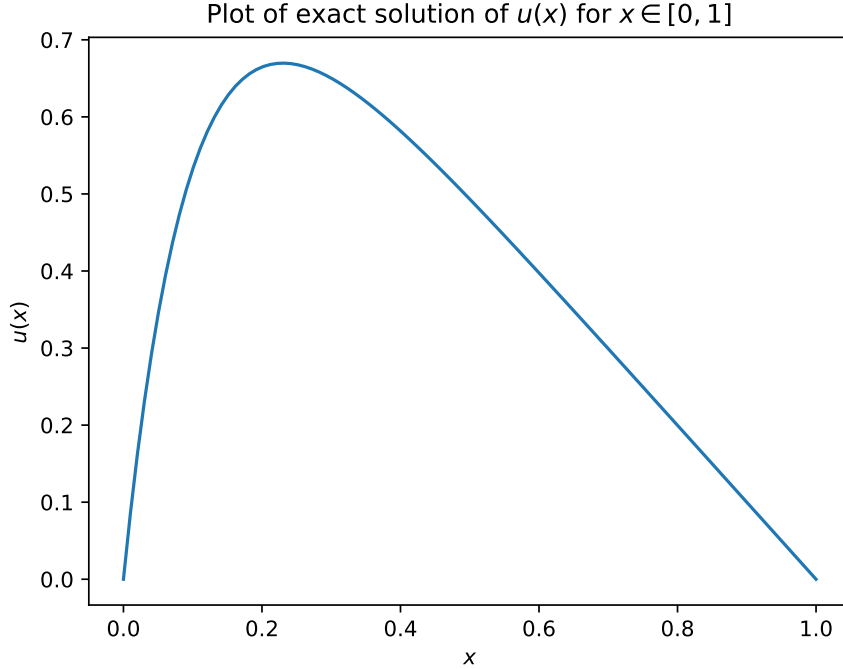


FIG. 1. Plot of the exact solution of $u(x)$ for $x \in [0, 1]$. We used 100 data points for the plot.

discretized version of (1), we will use $v(x)$ instead of $u(x)$ in the following procedures to make it clear that this is not an exact solution. The resulting equation becomes

$$\frac{-v(x - \Delta x) + 2v(x) - v(x + \Delta x))}{\Delta x^2} = -100e^{-10x} \quad (5)$$

Let's denote the i^{th} x -value in our set of data points with x_i , so that the corresponding v -value will be denoted with v_i . This implies that at the boundaries we have $x_0 = 0$, $v_0 = 0$, and $x_{N-1} = 1$, $v_{N-1} = 0$, where N are the number of data points. Thus, the final algorithm is

$$\frac{-v_{i-1} + 2v_i - v_{i+1}}{h^2} = f_i \quad (6)$$

where $h = \Delta x$ is defined as

$$h = \frac{x_N - x_0}{N - 1} \quad (7)$$

and we've rewritten the forcing term to include the minus sign that was originally on the left hand side of the equation, meaning that

$$f_i = f(x_i) = -100e^{-10x_i} \quad (8)$$

PROBLEM 4

Let us now multiply h^2 with both sides of (6), consequently rewriting the equation:

$$-v_{i-1} + 2v_i - v_{i+1} = g_i, \quad g_i = h^2 f_i \quad (9)$$

Furthermore, we may define the vectors \vec{v} and \vec{g} such that

$$\vec{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix}, \quad \vec{g} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{N-1} \end{bmatrix} \quad (10)$$

Using our knowledge of linear algebra, we may now define the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & \vdots \\ 0 & -1 & 2 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 2 \end{bmatrix} \quad (11)$$

which multiplied with \vec{v} gives us

$$\mathbf{A}\vec{v} = \begin{bmatrix} 2 & -1 & \dots & 0 \\ -1 & 2 & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & 2 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} 2v_0 - v_1 \\ -v_0 + 2v_1 - v_2 \\ \vdots \\ -v_{N-2} + 2v_{N-1} \end{bmatrix} \quad (12)$$

We immediately see from (9) that this gives us the vector \vec{g} containing the g_i -values with the same indexes as the corresponding v_i -values from \vec{v} . Thus, we can indeed rewrite the original discretized equation as the matrix equation

$$\mathbf{A}\vec{v} = \vec{g} \quad (13)$$

where \mathbf{A} is the tridiagonal matrix with subdiagonal, main diagonal and superdiagonal specified by the signature $(-1, 2, -1)$, just like we wanted to show.

PROBLEM 5

Problem 5a

We now have the vectors \vec{v}^* and \vec{x} , both of length m , and the matrix \mathbf{A} of size $n \times n$. This means that we have m number of data points, and that if we wish to solve (13) with the matrix \mathbf{A} we can only solve for n of the data points. This is because we never can multiply a matrix with a vector if the matrix has more or less columns than the amount of elements the vector has. Thus, the number n indicates how many of the values in the complete solution we will be able to calculate by solving the equation.

Problem 5b

If $n < m$ we will only be able to compute the first n elements in the complete solution \vec{v}^* , or alternatively the last n elements if we use the last n elements in \vec{x} to compute the elements in \vec{g} .

PROBLEM 6

Problem 6a

Now we consider the general tridiagonal matrix with subdiagonal, main diagonal and superdiagonal represented by the vectors \vec{a} , \vec{b} and \vec{c} , respectively. Such a matrix would have the form

$$\mathbf{A} = \begin{bmatrix} b_0 & c_0 & 0 & \dots & 0 \\ a_1 & b_1 & c_1 & \dots & \vdots \\ 0 & a_2 & b_2 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & a_{n-1} & b_{n-1} \end{bmatrix} \quad (14)$$

should its dimensions be $n \times n$. To study the general algorithm of computing (13), let us look at the case of $n = 3$. If we augment \vec{g} to \mathbf{A} so that we get

$$[\mathbf{A} \ \vec{g}] = \begin{bmatrix} b_0 & c_0 & 0 & g_0 \\ a_1 & b_1 & c_1 & g_1 \\ 0 & a_2 & b_2 & g_2 \end{bmatrix} \quad (15)$$

we can use the Gaussian elimination method to turn the part of the matrix that is originally just \mathbf{A} into the identity matrix. We could either use forward substitution or backwards substitution to go further, so let's try using the former. First, we can multiply row I of the augmented matrix with a_1/b_0 and then subtract that row from row II:

$$\begin{bmatrix} b_0 & c_0 & 0 & g_0 \\ 0 & b_1 - a_1c_0/b_0 & c_1 & g_1 - a_1g_0/b_0 \\ 0 & a_2 & b_2 & g_2 \end{bmatrix} \quad (16)$$

To make the calculations more readable, let's introduce the new variables $\bar{b}_1 = b_1 - a_1c_0/b_0$ and $\bar{g}_1 = g_1 - a_1g_0/b_0$. Furthermore, let's now multiply row II with a_2/\bar{b}_1 and subtract it from row III so that we're left with

$$\begin{bmatrix} b_0 & c_0 & 0 & g_0 \\ 0 & \bar{b}_1 & c_1 & \bar{g}_1 \\ 0 & 0 & b_2 - a_2c_1/\bar{b}_1 & g_2 - a_2\bar{g}_1/\bar{b}_1 \end{bmatrix} \quad (17)$$

Once again we introduce some new variables, $\bar{b}_2 = b_2 - a_2c_1/\bar{b}_1$ and $\bar{g}_2 = g_2 - a_2\bar{g}_1/\bar{b}_1$. Dividing row III with \bar{b}_2 we're then left with

$$\begin{bmatrix} b_0 & c_0 & 0 & g_0 \\ 0 & \bar{b}_1 & c_1 & \bar{g}_1 \\ 0 & 0 & 1 & v_2 \end{bmatrix} \quad (18)$$

What we're left with on the bottom left is now actually the third value in the solution to the original equation. To find v_2 and v_1 we work our way upward with similar calculations. First we multiply row III with c_1 and subtract it from row II, then we divide row II with \bar{b}_1 so that we have 1 on the diagonal element on row II. Finally we multiply row II with c_0 and subtract it from row I, then dividing row I with b_0 so that we're left with

$$\begin{bmatrix} 1 & 0 & 0 & v_0 \\ 0 & 1 & 0 & v_1 \\ 0 & 0 & 1 & v_2 \end{bmatrix} \quad (19)$$

As we can see we're now left with the vector \vec{v} that would satisfy (13) in column IV. This process is analogous for any equation where \mathbf{A} is an $m \times n$ matrix, \vec{v} is a vector of size n and \vec{g} is a vector of size m . A general pseudo code of this process when \mathbf{A} is a tridiagonal matrix could look like in Algorithm 1. Here it is assumed that one would first create a new matrix $\mathbf{A}g$ that is a copy of \mathbf{A} with \vec{g} concatenated to it.

Algorithm 1 Pseudo code for solving $\mathbf{A}\vec{v} = \vec{g}$ for \vec{v}

```

for  $i = 0, 1, \dots, n - 2$  do ▷ get rid of the subdiagonal
     $newrow = \mathbf{A}g_i * (\mathbf{A}g_{i+1,i} / \mathbf{A}g_{i,i});$  ▷ when there's only the first index we mean the whole row
     $\mathbf{A}g_{i+1} = \mathbf{A}g_{i+1} - newrow;$ 
for  $i = n - 1, n - 2, \dots, 1$  do ▷ get rid of the superdiagonal
     $\mathbf{A}g_i = \mathbf{A}g_i / \mathbf{A}g_{i,i};$ 
     $\mathbf{A}g_{i-1} = \mathbf{A}g_{i-1} - (\mathbf{A}g_{i-1,i} * \mathbf{A}g_i);$ 
 $\mathbf{A}g_0 = \mathbf{A}g_0 / \mathbf{A}g_{0,0};$  ▷ fix the first row so that the identity matrix is completed

```

Problem 6b

In Algorithm 1 we do 5 FLOPs for each time we loop over the first loop, and we do this $n - 1$ times. In the second loop we also do 5 FLOPs $n - 1$ times, and lastly we do one FLOP at the very end. This means that the total amount of FLOPs in this algorithm is $10 * (n - 1) + 1$.

TABLE I. Write a descriptive caption here, explaining the content of your table.

Number of points	Output
10	0.3086
100	0.2550

PROBLEM 7**Problem 7a****Problem 7b****PROBLEM 8****Problem 8a****Problem 8b****Problem 8c****PROBLEM 9****Problem 9a****Problem 9b****Problem 9c****PROBLEM 10**

We write equations using the LaTeX `equation` (or `align`) environments. Here is an equation with numbering

$$\mathbf{F} = \frac{d\mathbf{p}}{dt}, \quad (20)$$

and here is one without numbering:

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = 0.$$

Sometimes it is useful to refer back to a previous equation, like we're demonstrating here for equation 20.

Also, note the LaTeX code we used to get correct quotation marks in the previous sentence. (Simply using the " key on your keyboard will give the wrong result.) Figures should preferably be vector graphics (e.g. a .pdf file) rather than raster graphics (e.g. a .png file).

By the way, don't worry too much about where LaTeX decides to place your figures and tables — LaTeX knows more than we do about proper document layout. As long as you label all your figures and tables and refer to them in the text, it's all good. Of course, in some cases it can be worth trying to force a specific placement, to avoid the figure/table appearing many pages away from the main text discussing it, but this isn't something you should spend time on until the very end of the writing process.

Next up is a table, created using the `table` and `tabular` environments. We refer to it by table I.

Finally, we can list algorithms by using the `algorithm` environment, as demonstrated here for algorithm 2.

Algorithm 2 Some algorithm

Some maths, e.g. $f(x) = x^2$.

▷ Here's a comment

for $i = 0, 1, \dots, n - 1$ **do**

 Do something here

while Some condition **do**

 Do something more here

Maybe even some more math here, e.g. $\int_0^1 f(x)dx$
