FYS4150 - Project 1

github.com/isakrukan/FYS4150/Project1

Isak Cecil Onsager Rukan
(Dated: September 11, 2025)

The one-dimensional Poisson can be written as

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

where $f(x)$ is some known function. In this project we let $x \in [0, 1]$ and

$$f(x) = 100e^{-10x}, \tag{2}$$

together with the following boundary conditions

$$u(0) = 0, \quad u(1) = 0. \tag{3}$$

I. PROBLEM 1

Let $v(x) = d/dxu(x)$. Integrating (1) leads to

$$\begin{aligned} -v(x) &= 100 \int dx e^{-10x} \\ &= -10e^{-10x} + C, \end{aligned} \tag{4}$$

which means that

$$\begin{aligned} u(x) &= \int v(x) \\ &= -e^{-10x} - Cx + D. \end{aligned} \tag{5}$$

Here C and D are some constants which are determined by the boundary conditions in (3). Concretely, $u(0) = 0$ leads to $D = 1$ and then $u(1) = 0$ sets $C = -(1 - e^{-10})$. Hence,

$$u(x) = 1 - (1 - e^{-10x})x - e^{-10x}. \tag{6}$$

II. PROBLEM 2

Using problem2.cpp we compute $u(x)$ numerically for $N = 10^4$ points. The result is written to problem2.txt and Fig. 1 is made in problem2.py.

III. PROBLEM 3

The second derivative of $u(x)$, appearing on the R.H. side of (1) may be written as

$$\frac{d^2}{dx^2}u(x) = \lim_{h\to 0} \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}. \tag{7}$$

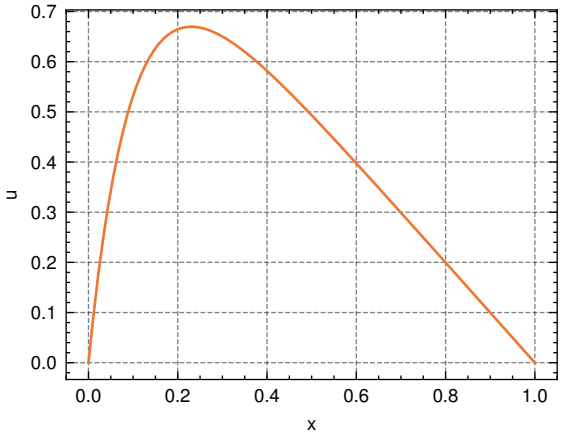

FIG. 1. Eq. (6) using $N = 10^4$ points.

To approximate this limit numerically, let h be small and $\vec{x}$ be an array of n points equispaced from zero to one. Eq. (7) can then be approximated up to order $\mathcal{O}(h^2)$:

$$\frac{d^2}{dx^2}u(x)|_{x_i} = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \mathcal{O}(h^2). \tag{8}$$

Since, according to (1), the second derivative of $u(x)$ equals $-f(x)$, we can write (8) as

$$-h^2 f(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + \mathcal{O}(h^2). \tag{9}$$

IV. PROBLEM 4

Let $\vec{v} = (v_1, ..., v_n)$ be a vector whose i-th component is thought of as representing the numerical implementation (so ignoring the order $\mathcal{O}(h^2)$) of u_{i+1}. Eq. (9) can then be written as

$$-v_{i-1} + 2v_i - v_{i+1} = h^2 f(x_i). \tag{10}$$

Further, let $\vec{g}$ be the vector of length n, given by componentwise as

$$\vec{g}_i = h^2 f(x_{i+1}), i = 2, ..., n-1, \tag{11}$$

and $\vec{g}_1 = h^2 f(x_1) + u(0)$, $\vec{g}_n = h^2 f(x_n) + u(1)$. Then, if we let A be the $n \times n$-matrix given by

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}, \tag{12}$$

we can represent (10) as

$$A\vec{v} = \vec{g}. \tag{13}$$

To see that this explicitly, multiply the matrix A with $\vec{v}$, which gives $A\vec{v}$ componentwise as

$$(A\vec{v})_i = -v_{i-1} + 2v_i - v_{i+1}, \quad i = 2, ..., n-1. \tag{14}$$

For $i = 0$: $(A\vec{v})_0 = 2v_0 - v_1$, and $i = n$: $(A\vec{v})_n = -vn - 1 + 2v_n$.

V. PROBLEM 5

The vector $\vec{v}$ does not contain the boundary points corresponding to $x_0 = 0$ and $x_{N-1} = 1$. Let the vector $\vec{v}^*$ be given by

$$\vec{v}^* = (u(0), \vec{v}, u(1)). \tag{15}$$

Then, clearly, $\vec{v}^*$ is a vector of length n and when solving (13) we will obtain $\vec{v}$, which is $\vec{v}^*$ excluding its first and last component.

VI. PROBLEM 6

Consider now A as a general tridiagonal matrix with vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ representing the subdiagonal, main diagonal and superdiagonal, respectively. When solving (13) we may then use a number of row operations on A, such that it turns into a matrix $\tilde{A}$ with all entries below the diagonal equal to zero. These row operations naturally alters the $\vec{g}$ on R.H. side of (13) into a vector $\tilde{g}$, at which point the solution to (13) can then be simply read of by multiplying out $\tilde{A}\vec{v} = \vec{\tilde{g}}$.

Let R_i denote the i-th row of (13). In the following, when writing $\overset{aR_i - bR_j}{\rightarrow}$, we mean the row operation of subtracting b times row j from a times row i. When applying a row operation to the i-th row, we write $\tilde{b}_j$ and $\tilde{g}_j$ as the resulting new j-th component of the vector $\vec{b}$ and $\vec{g}$, respectively. We have:

$$\left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ R_2: & a_2 & b_2 & c_2 & \ddots & \vdots & g_2 \\ R_3: & 0 & a_3 & b_3 & \ddots & 0 & g_3 \\ & \vdots & \ddots & \ddots & \ddots & c_{n-1} & \vdots \\ R_n: & 0 & \cdots & 0 & a_{n-1} & b_n & g_n \end{array}\right]$$

$$\overset{R_2 - a_2/b_1 R_1}{\longrightarrow} \left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ R_2: & 0 & b_2 - a_2\frac{c_1}{b_1} & c_2 & \ddots & \vdots & g_2 - g_1\frac{a_2}{b_1} \\ R_3: & 0 & a_3 & b_3 & \ddots & 0 & g_3 \\ & \vdots & \ddots & \ddots & \ddots & c_{n-1} & \vdots \\ R_n & 0 & \cdots & 0 & a_{n-1} & b_n & g_n \end{array}\right]$$

$$\overset{R_3 - a_3/\tilde{b}_2 R_2}{\longrightarrow} \left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ R_2: & 0 & \tilde{b}_2 & c_2 & \ddots & \vdots & \tilde{g}_2 \\ R_3: & 0 & 0 & b_3 - a_3\frac{c_2}{\tilde{b}_2} & \ddots & 0 & g_3 - g_2\frac{a_3}{\tilde{b}_1} \\ & \vdots & \ddots & \ddots & \ddots & c_{n-1} & \vdots \\ R_n & 0 & \cdots & 0 & a_{n-1} & b_n & g_n \end{array}\right] \tag{16}$$

Continuing in this way we obtain

$$\tilde{b}_j = b_j - a_j \frac{c_{j-1}}{b_{j-1}}, \quad j = 2, ..., n, \tag{17a}$$

$$\tilde{g}_j = g_j - a_j \frac{g_{j-1}}{b_{j-1}}, \quad j = 2, ..., n, \tag{17b}$$

with $\tilde{b}_1 = b_1$ and $\tilde{g}_1 = g_1$. After having performed these row operations, we can perform backward substitution from the last row:

$$\begin{array}{c} \\ \xrightarrow{R_n/\tilde{b}_n} \end{array} \left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ R_{n-2}: & 0 & \cdots & \tilde{b}_{n-2} & c_{n-2} & 0 & \tilde{g}_{n-2} \\ R_{n-1}: & 0 & \cdots & 0 & \tilde{b}_{n-1} & c_{n-1} & \tilde{g}_{n-1} \\ R_n: & 0 & \cdots & 0 & 0 & \tilde{b}_n & \tilde{g}_n \end{array}\right] \\ \left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ R_{n-2}: & 0 & \cdots & \tilde{b}_{n-2} & c_{n-2} & 0 & \tilde{g}_{n-2} \\ R_{n-1}: & 0 & \cdots & 0 & \tilde{b}_{n-1} & c_{n-1} & \tilde{g}_{n-1} \\ R_n: & 0 & \cdots & 0 & 0 & 1 & \frac{\tilde{g}_n}{\tilde{b}_n} \end{array}\right] \tag{18}$$

meaning that

$$v_n = \frac{\tilde{g}_n}{\tilde{b}_n}. \tag{19}$$

Then, performing the row-operation

$$R_{n-1} \to (R_{n-1} - c_{n-1} R_n)/\tilde{b}_{n-1}, \tag{20}$$

gives

$$\begin{array}{c} \\ \xrightarrow{\frac{R_{n-2}}{\tilde{b}_{n-2}} - \frac{c_{n-2}}{\tilde{b}_{n-2}} R_{n-1}} \end{array} \left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ R_{n-2}: & 0 & \cdots & \tilde{b}_{n-2} & c_{n-2} & 0 & \tilde{g}_{n-2} \\ R_{n-1}: & 0 & \cdots & 0 & 1 & 0 & \frac{\tilde{g}_{n-1} - c_{n-1} v_n}{\tilde{b}_{n-1}} \\ R_n: & 0 & \cdots & 0 & 0 & 1 & v_n \end{array}\right] \\ \left[\begin{array}{cccccc|c} R_1: & b_1 & c_1 & 0 & \cdots & 0 & g_1 \\ & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ R_{n-2}: & 0 & \cdots & 1 & 0 & 0 & \frac{\tilde{g}_{n-2} - c_{n-2} v_{n-1}}{\tilde{b}_{n-2}} \\ R_{n-1}: & 0 & \cdots & 0 & 1 & 0 & v_{n-1} \\ R_n: & 0 & \cdots & 0 & 0 & 1 & v_n \end{array}\right], \tag{21}$$

and so on, which leads to

$$v_i = \frac{\tilde{g}_i - c_i v_{i+1}}{\tilde{b}_i}, \quad i = n-1, ..., 1. \tag{22}$$

Eq. (17a) and (17b) requires 3 and 7 floating-point operations (FLOPs) each, and hence implementing (17) takes a total of $(3+5)*(n-1)$ FLOPs. (Here, we have counted computing $\tilde{g}_j$ as 2 FLOPs; counting h^2 as a single FLOP (since this can be done beforehand) and evaluating f as a single FLOP (though that is technically wrong but dependent on the realization of f).) Eq. (19) takes one FLOP, while (22) takes $3*(n-1)$ FLOPs. Hence, the complete algorithm for obtaining $\vec{v}$ requires $8*(n-1)+3*(n-1)+1 = 11n-10 = \mathcal{O}(n)$ FLOPs.

VII. PROBLEM 7

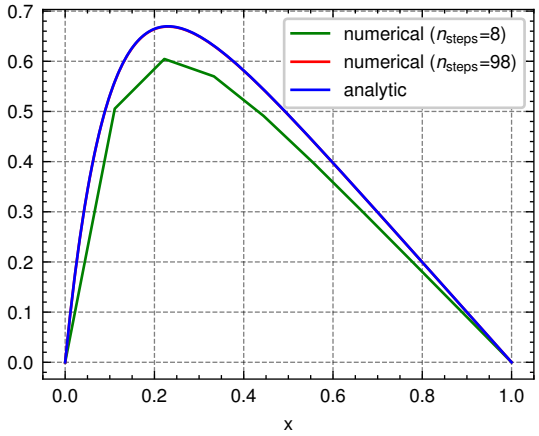

FIG. 2. Numerically computed solutions to the Poisson equation (1) (with source term given in (2)), using the general algorithm described in section VI for $n_{\text{steps}} = 8$ (green) and $n_{\text{steps}} = 98$ (red). The analytic solution in (6) is shown in blue.

In problem7.cpp the algorithm described in section VI, which we refer to as the "general algorithm", is implemented and solves the Poisson equation using $n_{\text{steps}} \in 10, 100, 1000, 100000$, writing to the file problem7.txt. Using the computed values for $\vec{v}$, Fig. 2 is created in problem7.py.

Fig. 2 shows that the numerically computed solution to the Poisson equation with $f(x)$ given in (2) quite quickly, seemingly, converges to the analytic solution. Already at $n_{\text{steps=100}}$ there is no visual difference between the numerical and analytic solution.

VIII. PROBLEM 8

We will now consider the absolute error, which is defined componentwise as

$$(\epsilon_{\text{abs}})_i = \log_{10} |\vec{u}_i - \vec{v}_i|, \tag{23}$$

and the relative error:

$$(\epsilon_{\text{rel}})_i = \log_{10} \left| \frac{\vec{u}_i - \vec{v}_i}{\vec{u}_i} \right|. \tag{24}$$

Fig. 3 shows the absolute error (23) and fig. 4 shows the relative error (24), both computed in problem8.py, using the values computed in section VII.

In both figures, the boundary points $u(0) = u(1) = 0$ are excluded. The absolute error in Fig. 3 is mostly constant and decreasing with increasing n_{steps}, though converges to zero at the $x = 0$ and $x = 1$. This should be expected for any descent numerical approximation, as the analytic solution vanishes at these end points.

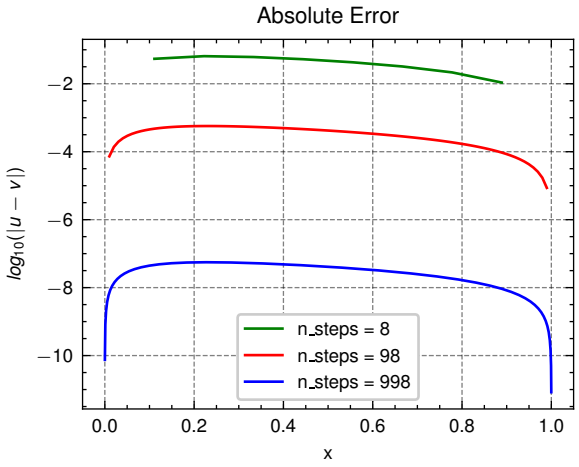

FIG. 3. The absolute error between the numerical approximation to the Poisson equation (1) (with source term given in (2)) using the general algorithm described in section VI for $n_{\text{steps}} = 8$ (green), $n_{\text{steps}} = 98$ (red) and $n_{\text{steps}} = 998$ (blue).

The relative error in Fig. 4 is almost perfectly constant for all $x \in (0, 1)$. This supplements the behavior of the absolute error (which is also mostly constant), but also shows that the divergence of the numerical solution from the analytic solution remains constant even when the analytic solution goes to zero at the endpoints.

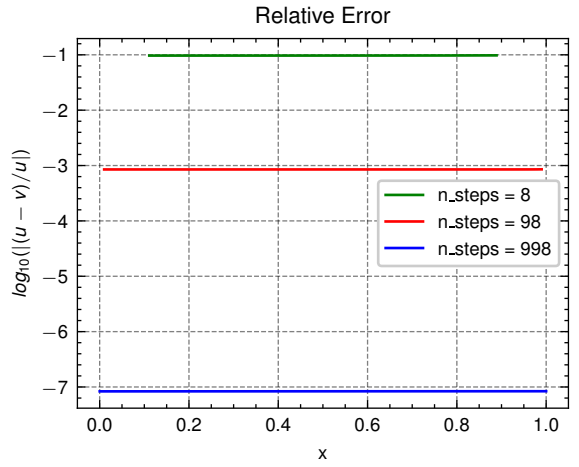

FIG. 4. The relative error between the numerical approximation to the Poisson equation (1) (with source term given in (2)) using the general algorithm described in section VI for $n_{\text{steps}} = 8$ (green), $n_{\text{steps}} = 98$ (red) and $n_{\text{steps}} = 998$ (blue).

Table VIII shows the maximum absolute and relative errors for $n_{\text{steps}} = 10^1, 10^2, ..., 10^7$. The table indicates that there is a turning point at $n_{\text{steps}} \approx 10^5$ where the behavior, which prior to $n_{\text{steps}} \approx 10^5$ is that the errors are decreasing for increasing n_{steps}, starts to increase.

Maximum Absolute and Relative Errors		
Absolute error ($\log_{10}$)	Relative error ($\log_{10}$)	n_{steps}
-1.19	-1.01	10^1
-3.24	-3.07	10^2
-5.25	-5.08	10^3
-7.25	-7.08	10^4
-9.04	-8.84	10^5
-6.35	-6.08	10^6
-6.20	-5.53	10^7

TABLE I. The maximum absolute and relative errors for $n_{\text{steps}} = 10^1, 10^2, ..., 10^7$.

IX. PROBLEM 9

The general algorithm from section VI can be specialized since we know that $\vec{b}_i = 2$ and $\vec{c}_i = \vec{a}_i = -1$. Concretely, (17) can be written as

$$\tilde{b}_i = 2 - \frac{1}{b_{i-1}}, \quad i = 2, ..., n, \tag{25a}$$

$$\tilde{g}_i = h^2 f(x_{i+1}) + \frac{g_{i-1}}{b_{i-1}}, \quad i = 2, ..., n, \tag{25b}$$

whilst the forward substitution process which led to (22), can be written as

$$v_i = \frac{\tilde{g}_i + v_{i+1}}{\tilde{b}_i}, \quad i = n-1, ..., 1, \tag{26}$$

with $v_n = \tilde{g}_n/\tilde{b}_n$. Implementing (25) requires $(2+4)*(n-1)$ FLOPs, while (26) requires $3*(n-1)$ (again, counting evaluating f as a single FLOP), and setting $v_n = \tilde{g}_n/\tilde{b}_n$ requires one. Hence, the total number of FLOPs for the specialized algorithm is $(2+4)*(n-1)+3*(n-1)+1 = 9n-8$.

X. PROBLEM 10

The special algorithm in section IX requires fewer FLOPs than that of the general algorithm in section VI. Table X shows the averaged time duration (over 100 runs) of running the general/special algorithm.

Averaged Time Duration of Special/General Algorithm		
Special Algorithm [s]	General Algorithm [s]	n_{steps}
$1.06 \cdot 10^{-4}$	$3.00 \cdot 10^{-5}$	10^1
$2.12 \cdot 10^{-4}$	$1.18 \cdot 10^{-4}$	10^2
$1.64 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	10^3
$2.18 \cdot 10^{-2}$	$1.37 \cdot 10^{-2}$	10^4
$2.04 \cdot 10^{-1}$	$1.26 \cdot 10^{-1}$	10^5
$1.94 \cdot 10^{0}$	$1.17 \cdot 10^{0}$	10^6

TABLE II. The averaged time duration over 100 runs of the special (from section VI) versus the general (from section IX) algorithm.