

# FYS3150 - project 1

Sverre Wehn Noremsaune & Frida Marie Engøy Westby  
(Dated: September 12, 2021)

<https://github.uio.no/comPhys/FYS3150/tree/project1>

## PROBLEM 1

We have the one-dimensional Poisson equation

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

where  $f(x)$  is known to be  $100e^{-10x}$ . We also assume  $x \in [0, 1]$ , that the boundary condition are  $u(0) = 0 = u(1)$  and  $u(x)$  is

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

where  $u(x)$  is an exact solution to Eq. (1). We can check this analytically by differentiating  $u(x)$  twice.

$$\begin{aligned} -u''(x) &= f(x) \\ u''(x) &= -f(x) \\ u(x)' &= 10x^{-10x} - 1 + \frac{1}{e} \\ u''(x) &= -100e^{-10x} = -f(x) \quad \blacksquare \end{aligned}$$

## PROBLEM 2

### problem a)

See 'poisson\_exact.cpp' in the github repository

### problem b)

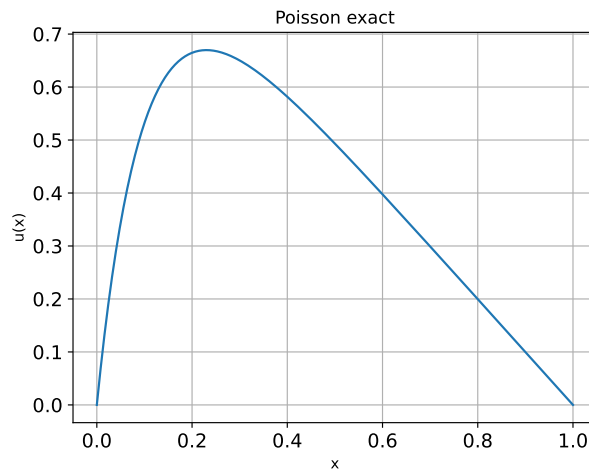


FIG. 1. A plot of the exact solution for the Poisson equation from 1 for  $x \in [0, 1]$

### PROBLEM 3

We are discretizing the Poisson equation from 1.  
Discretizing x and setting up some notation:

$$\begin{aligned}x &\rightarrow x_i \\ u(x) &\rightarrow u_i \\ i &= 0, 1, \dots, n \\ h &= \frac{x_{max} - x_{min}}{n} \\ x_i &= x_0 + ih\end{aligned}$$

We're using the three-point formula to find the second derivative:

$$\frac{du^2}{dx^2} = u'' = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + O(h^2)$$

$$f_i = - \left( \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + O(h^2) \right)$$

We then approximate and change the notation,  $v_i \approx u_i$  and get

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

### PROBLEM 4

The equation we got in 3 isn't the most ergonomic for setting up a matrix equation, so we can rewrite it to

$$-v_{i-1} + 2v_i - v_{i+1} = h^2 f_i \quad (4)$$

We know:  $f_i, v(0) = v(1) = 0$  4 is a set of equations for every i

$$\begin{aligned}i = 1 & \quad -v_0 \quad 2v_1 \quad -v_2 &= h^2 f_i \\ i = 2 & \quad \quad -v_1 \quad 2v_2 \quad -v_3 &= h^2 f_2\end{aligned}$$

and so on and so forth. We can see that  $v_0$  and  $v_n$  will end up and alone on their columns, and since we know what they are we simply move them over.

$$\begin{aligned}i = 1 & \quad 2v_1 \quad -v_2 &= h^2 f_i + v_0 \\ i = 2 & \quad -v_1 \quad 2v_2 \quad -v_3 &= h^2 f_2\end{aligned}$$

This can then be easily rewritten as a matrix equation  $A\vec{v} = \vec{g}$

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -1 & 2 & -1 & 0 \\ 0 & 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & 0 & \dots & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{n-3} \\ v_{n-2} \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ gn-3 \\ gn-2 \\ gn-1 \end{bmatrix}$$

where  $g_i$  is  $h^2 f_i$  ( $+v_0$  for  $f_1$  and  $+v_n$  for  $f_{n-1}$ )

**PROBLEM 5****Problem a**

Since we're "dropping" 2 columns,  $n = m - 2$

**problem b**

we will find  $\vec{v}_i^*$  for  $1..(n-1)$ , meaning everything but the boundary points.

**problem 6****problem a**


---

**Algorithm 1** Algorithm for solving general tridiagonal matrix
 

---

arrays a, b, c, u, f, temp of length n

btemp = b[1]

u[1] = f[1]/btemp

**for** i = 2,3,...,n **do**

temp[i] = c[i-1] / btemp

btemp = b[i] - a[i] \* temp[i]

u[i] = (f[i] - a[i] \* u[i-1]) / btemp

**for** i = n-1, n-2, ..., 1 **do**

u[i] -= temp[i+1] \* u[i+1]

---

**problem b**

FLOPs:  $1 + 6(n-1) + 2(n-1) = 1 + 8(n-1) = 8n - 7$

**PROBLEM 7****problem a**

see 'general\_tridiag.cpp' in the github repository

problem b

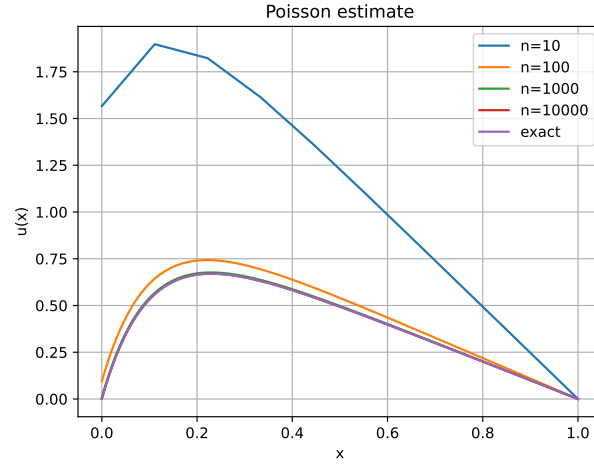


FIG. 2. Exact vs numerical comparison for the solution of equation 1

PROBLEM 9

problem a

problem b

problem c

PROBLEM 9

Since the values of  $a, b, c$  never changes, we can just replace the arrays with a constants, saving us from repeatedly calculating  $-(-1) \cdot something$ .

problem a

---

**Algorithm 2** Algorithm for solving special tridiagonal matrix

---

arrays  $u, f$ , temp of length  $n$

btemp = 2

$u[1] = f[1] / 2$

**for**  $i = 2, 3, \dots, n$  **do**

temp[i] = -1 / btemp

btemp =  $b[i] + temp[i]$

$u[i] = (f[i] + u[i-1]) / btemp$

**for**  $i = n-1, n-2, \dots, 1$  **do**

$u[i] -= temp[i+1] * u[i+1]$

---

problem b

FLOPs:  $1 + 4(n-1) + 2(n-1) = 1 + 6(n-1) = 6n - 5$

problem c