

Title of the document

Henrik Modahl Breitenstein and Carl Petter Duedahl
(Dated: September 11, 2021)

<https://github.com/henrikbreitenstein/FYS3150.git>

PROBLEM 1

Poisson likningen

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

Bytter $f(x)$ med gitt funksjon

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

Tar integralene

$$\begin{aligned} -\iint d^2u &= \iint 100e^{-10x} dx^2 \\ -u &= \int -10e^{-10x} + c_1 dx \\ -u &= e^{-10x} + c_1x + c_2 \\ u &= -e^{-10x} - c_1x - c_2 \end{aligned}$$

Bruker initialbetingelsene

$$u(0) = 0 \Rightarrow -1 - c_2 = 0 \quad (1)$$

$$u(1) = 0 \Rightarrow -e^{-10} - c_1 - c_2 = 0 \quad (2)$$

Med **1** og **2** får vi:

$$\begin{aligned} c_2 &= -1 \\ c_1 &= 1 - e^{-10} \end{aligned}$$

Ved å sette inn for c_1 og c_2 får vi:

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

PROBLEM 2

I repository'et under FYS3150/Project1/main.cpp så har vi skrevet koden som regner ut verdiene til den eksakte løsningen, og i FYS3150/Project1/plot.py så tegnes grafen. Kan kjøre programmet 'main.cpp' med commandoen

```
$ make all  
og kjøre 'plot.py' med
```

```
$ python plot.py
```

PROBLEM 5

Problem a

Siden vi gjør en matrisemultiplikasjon så må bredden, n , til \mathbf{A} være det samme som lengden m til \vec{v} og \vec{g} . Så

$$n = m$$

Problem b

Av $A\vec{v} = \vec{g}$ vil vi finne alle verdier mellom grensebetingelsene $u(0) = u(1) = 0$. \vec{v}^* er den samme som \vec{v} men vi har lagt til v_0 og v_{n+1} som er grensebetingelsene.

PROBLEM 6

A

Vi har nå en vektor

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

og en $n \times n$ -matrise

$$\mathbf{A} = \begin{pmatrix} b_1 & c_2 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & a_n & b_n \end{pmatrix}$$

Matrisemultipliserer vi disse får vi

$$\begin{aligned} \mathbf{A}\vec{v} &= \begin{aligned} (I) \quad & b_1 v_1 + c_1 v_2 & & & = g_1 \\ (II) \quad & a_2 v_1 + b_2 v_2 + c_3 v_3 & & & = g_2 \\ (III) \quad & & a_3 v_2 + b_3 v_3 + c_3 v_4 & & = g_3 \\ & \vdots & & & \\ (n) \quad & & & a_n v_{n-1} + b_n v_n & = g_n \end{aligned} \end{aligned}$$

Herfra skal vi radredusere og starter med å ta $(II) - \frac{a_2}{b_1}(I)$ slik at vi får

$$(II^*) \quad 0 \quad b_2 - \frac{c_1 \cdot a_2}{b_1} \quad c_2 \quad \cdots \quad g_2 - g_1 \frac{a_2}{b_1}$$

og vi ser da at a_2 går bort. Vi kan også sette $b_2^* \equiv b_2 - \frac{a_2 \cdot c_1}{b_1}$ og $g_2^* \equiv g_2 - g_1 \frac{a_2}{b_1}$. Da har vi mellom (II^*) og (III) noe som ser ganske likt ut som det vi hadde mellom (I) og (II) . Derfor gjør vi det samme som vi gjorde før og tar $(III) - \frac{a_3}{b_2^*}(II)^*$ og får

$$(III)^* \quad 0 \quad 0 \quad b_3 - \frac{c_2 \cdot a_3}{b_2^*} \quad c_3 \quad g_3 - g_1 \frac{a_3}{b_2^*}$$

og vi kan igjen definere $b_3^* = b_3 - \frac{c_2 \cdot a_3}{b_2^*}$ og $g_3^* = g_3 - g_1 \frac{a_3}{b_2^*}$. Og vi kan da fortsette med dette nedover som $(k) - \frac{a_k}{b_{k-1}^*}(k-1)^*$. Så har vi fjernet a -ene så da må vi fjerne c -ene og normalisere b -ene. For å gjøre starter vi i siste ledd med

$$(n)/b_n$$

så

$$b_n = \frac{b_n}{b_n}$$

og

$$g_n^* = \frac{g_n}{b_n}$$

Da får vi at $A_n v_n = g_n$ blir satt ned til

$$v_n = \frac{g_n}{b_n}$$

$$\begin{pmatrix} (n-1)^* & 0 & \cdots & b_{n-1}^* & c_{n-1} & g_{n-1}^* \\ (n)^* & 0 & \cdots & 0 & b_n^* & g_{n-1}^* \end{pmatrix}$$

så hvis vi da tar $(n-1)^* - \frac{c_{n-1}}{b_n^*}(n)^*$ får vi

$$\begin{pmatrix} \tilde{n-1} & 0 & \cdots & b_{n-1}^* & 0 & g_{n-1}^* - \frac{c_{n-1}g_n^*}{b_n^*} \end{pmatrix}$$

og vi setter $g_{n-1}^* = g_{n-1}^* - \frac{c_{n-1}g_n^*}{b_n^*}$ og dette gjør vi videre oppover som $(k-1) - \frac{c_k}{b_{k-1}^*}(k)$ Til slutt står vi da bare igjen med b^* -ene og disse kan vi da dele på seg selv og vi får en identitetsmatrise. Vi kan nå skrive dette som en algoritme. Anta vi har en tridiagonal matrise

$$A = U = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n,1} & u_{n,2} & \cdots & u_{n,n} \end{pmatrix}$$

og en som skal løses for vektoren

$$g = h = \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{pmatrix}$$

Algorithm 1 Radredusering av tridiagonal matrise**for** $i = 2, 3, \dots, n$ **do**▷ Forward substitution, $n - 1$ repetisjoner

$$t \leftarrow \frac{u_{i,i-1}}{u_{i,i-1} - 1}$$

▷ 1 FLOP

$$u_{i,i-1} \leftarrow u_{i,i-1} - u_{i-1,i-1} \cdot t$$

▷ 2 FLOPs

$$u_{i,i} \leftarrow u_{i,i} - u_{i-1,i-1} \cdot t$$

▷ 2 FLOPs

$$h_i \leftarrow h_i - h_{i-1}t$$

▷ 2 FLOPs

for $j = n - 1, n - 2, \dots, 1$ **do**▷ Til sammen $7 \cdot (n - 1)$ FLOPs i loopen
▷ Backward Substitution, $n - 1$ repetisjoner

$$k \leftarrow \frac{u_{j,j+1}}{u_{j+1,j+1}}$$

▷ 2 FLOPs

$$u_{j,j+1} \leftarrow u_{j,j+1} - u_{j+1,j+1} \cdot k$$

▷ 2 FLOPs

$$u_{j,j} \leftarrow u_{j,j} - u_{j+1,j} \cdot k$$

▷ 2 FLOPs

$$h_j \leftarrow h_j - h_{j+1} \cdot k$$

▷ 2 FLOPs

for $l = 1, 2, \dots, n$ **do**▷ Til sammen $7 \cdot (n - 1)$ FLOPs i loopen
▷ Deler for å få identitetsmatrise, til sammen n repetisjoner

$$u_l \leftarrow \frac{u_l}{u_l}$$

▷ 1 FLOP

$$h_l \leftarrow \frac{h_l}{u_l}$$

▷ 1 FLOPs

▷ Til sammen $2 \cdot n$ FLOPs

Da ser vi at vi til sammen får $2 \cdot n + 2 \cdot 7 \cdot (n - 1) = 16n - 14$ FLOPs.

PROBLEM 7

Sriptet 'Problem7new.cpp' og 'Problem7func.cpp' bruker den generelle algoritmen til å løse matriselikningen. For å kjøre de sammen:

\$ make pr7all

PROBLEM 8

Problem a

problem b

Problem c

PROBLEM 9

Algorithm 2 Spesialisert algoritme

	$a \leftarrow -1$	
	$b \leftarrow 2$	
	$c \leftarrow -1$	
	$g \leftarrow \vec{0}_n$	
	$\tilde{b} \leftarrow \vec{0}_n$	
	$\tilde{g} \leftarrow \vec{0}_n$	
	$v \leftarrow \vec{0}_n$	
	$\tilde{b}_0 \leftarrow b$	
for $i = 0, 1, 2, \dots, n$ do		▷ Fyller g med verdiene til $f(x)$
	$g_i \leftarrow f(x_i)$	
	$\tilde{g}_0 \leftarrow g_0$	
	$d \leftarrow a \cdot c$	
for $i = 0, 1, 2, \dots, n-1$ do		▷ Forward elimination
	$\tilde{b}_{i+1} \leftarrow \tilde{b}_{i+1} - \frac{d}{\tilde{b}_i}$	
		▷ 2 FLOPs
	$\tilde{g}_{i+1} \leftarrow g_{i+1} - \frac{a}{\tilde{b}_i} g_i$	
		▷ 3 FLOPs
for $i = n-1, n-2, \dots, 0$ do		▷ Backward elimination
	$v_i \leftarrow \frac{\tilde{g}_i - cv_{i+1}}{\tilde{b}_i}$	
		▷ 3 FLOPs

Problem a

problem b

Problem c

I filen 'solve.cpp' har vi kodet den spesielle algoritmen. Kjøres ved kommandoen

\$ make solveall