Technical University of Denmark

2. Obligatoriske Aflevering

02601 – Introduktion til numeriske algoritmer

Forår 2015

Udearbejde af:	*Ansvarsområde*
Ran Wang – s111503	*Spørgsmål:4,5,6, C (English)*
Alireza Nedaei – s113681	*Resten af afleveringen:50%*
Mathias Thage Hansen – s113680	*Resten af afleveringen:50%*

1. OPTÆLLING AF FLOPS

1.1

Beregningen $s = s + x(i)$ bruger 1 flop for hver gang det bliver beregnet. Denne beregning bliver så gjort det samme antal gange som der er værdier i x-vektoren så denne beregning bruger 6 flops til at beregne summen af s.

1.2

Til at beregne g bruges der 1 flop ved hver gang $g = g + x(i)$ og denne beregning gøres n gange. Så til for funktionen bruges n flops. Derudover bliver det lavet et flop ved $g = \frac{g}{n}$. Så der skal bruges n+1 flops til at løse for g.

For at beregn RMS der bruges 2 flops hver gang $RMS = RMS + x(i)^2$ da plus er et flop og $x(i)^2$ tæller for 1 flop. Dette gøres n gange hvilket giver for for funktionen 2*n flops. Derefter laves beregningen$RMS = sqrt\left(\frac{RMS}{n}\right)$ hvor kvadratroden tæller for en flop og divisionen tæller for en flop hvilket giver 2 flops så denne beregning kræver 2*n+2 flops.

Den samlet beregning kræver derfor 3*n+3 flops at lave denne beregning.

1.3

Antallet af flops der er nødvendigt til at lave elemineringen i gauss-eliminationen er følgende. I linje 2 $\text{factor} = \text{A(i} + 1, \text{i)} / \text{A(i, i)}$ er der en division hvilket giver et flop og i linje 3 er der ikke nogen flops. I linje 4 $\text{A(i} + 1, \text{i} + 1) = \text{A(i} + 1, \text{i} + 1) - \text{factor} * \text{A(i, i} + 1)$ er der både en multiplikation og en subtraktion hvilket giver 2 flops. I linje 5 $\text{b(i} + 1) = \text{b(i} + 1) - \text{factor} * \text{b(i)}$ er der også en multiplikation og en subtraktion hvilket også giver 2 flops. Dette giver i alt 5 flops.

For funktionen der genneløber disse beregner bliver så gennemført n-1 gange hvilket giver det totale antal flops til at være $5 * (n - 1) = 5n - 5$ flops.

2. SKRUER OG MØTRIKKER

2.1

Til at lave beregningen af masserne af skruer, møtrikker og søm opskrives en A-matrice med antallet af hvor mange af de forskellige ting hver person har. Hver cologne er hvor mange skruer, møtrikker og søm som hver person har. Den første cologne er for Anders, den midterste er for Birger og den sidste er for Carl.

$$\begin{matrix} skruer: \\ møtrikker: \\ søm: \end{matrix}\begin{bmatrix} 3 & 12 & 10 \\ 12 & 0 & 20 \\ 0 & 2 & 30 \end{bmatrix} = A$$

Der laves også en B matrice som fortæller masserne de forskellige har vejet deres antal af skruer, møtrikker og søm til.

$$\begin{matrix} Anders \\ Birger \\ Carl \end{matrix}\begin{bmatrix} 72{,}3 \\ 99{,}5 \\ 56{,}6 \end{bmatrix} = b$$

Så kommer et lineære ligningssystem til at se ud som følgende:

$$\begin{bmatrix} 3 & 12 & 10 \\ 12 & 0 & 20 \\ 0 & 2 & 30 \end{bmatrix} * \begin{bmatrix} m_{skruer} \\ m_{møtrik} \\ m_{søm} \end{bmatrix} = \begin{bmatrix} 72{,}3 \\ 99{,}5 \\ 56{,}6 \end{bmatrix}$$

2.2

Der er så benyttet Matlabs funktion ”backslash” til at beregne de forskellige masser. Massen af en skrue blev 5,51 g, massen af en møtrik blev 3,26 g og massen af et søm blev 1,67 g.

2.3

Efter at de tre personer vejer deres skruer, møtrikker og søm igen kommer der en ny b matrice.

$$\begin{bmatrix} 73{,}3 \\ 98{,}4 \\ 57{,}1 \end{bmatrix} = b$$

Dette giver et nyt lineært ligningssystem:

$$\begin{bmatrix} 3 & 12 & 10 \\ 12 & 0 & 20 \\ 0 & 2 & 30 \end{bmatrix} * \begin{bmatrix} m_{skruer} \\ m_{møtrik} \\ m_{søm} \end{bmatrix} = \begin{bmatrix} 73{,}3 \\ 98{,}4 \\ 57{,}1 \end{bmatrix}$$

Så løses dette ligningssystem på samme måde som i delopgave 2 med ”backslash” funktionen. Dette giver resultatet at massen af en skrue blev 5,40g, massen af en møtrik blev 3,36g og massen af et søm blev 1,68g.

2.4

Til at finde den største fejl som vægten laver i forhold til den oprindelige vægt vælges først at finde den absolutte fejl difference. Derefter benyttes funktionen max i Matlab til at bestemme den største fejl og i hvilken position den er i. **Den største fejl er på 0,11 i den første værdi hvilket er masse for skruer. Dette er meget logisk da det er dem der har den største masse.**

3. BEREGNINGSTIDER

3.1

Opstilling af funktion kan ses på figur 1

```
%%3.1 opstilles en funktion
function [t,backslash,inverse,x]=beregningstider(n)
A=rand(n)/5;
b=rand(n,1);
tic
backslash=A\b;%den første metod backlash
t(1)=toc;
tic
inverse = inv(A)*b; %2.invers
t(2)=toc;
tic
D=det(A);
C=A;
x=zeros(n,1); %cramers metod
for i=1:n
    C(:,i)=b;
    x(i)=det(C)/D;
    C(:,i)=A(:,i);
end
t(3)=toc;
end
```

Figur 1: forskellige metoder til bestemmelse af x

3.2

Ved at gøre bruge af beregningstiderne, opstilles nedstående tabel for n=100, 200, 400, 800

n	Backslash	Inverse	Crame
100	5,2877723e-04	8,6184611e-04	1,9933483e-02
200	1,1499385e-03	2,1126778e-03	1,6055298e-01
400	6,8992260e-03	1,6024989e-02	2,1914599e+00
800	4,9946960e-02	1,1064451e-01	2,7638978e+01

Ifølge bogen side 220, er det hurtigste at benytte metoden med 'backslash' hvilket passer fint med resultater fra overstående tabel. Dernæst kommer Inverse og til sidst og den klart langsomste metode er når man bruger Crame da den beregner det vha. determinanter. Hermed sammenlignes for n=800 i forhold til de to andre metoder. Det ses at backslash er 0.06 sekunder hurtigere end inverse, trods 27.5891 sekunder hurtigere end at benytte Crame metoden. Dermed beregnes på den procentvis afvigelse som er ca. 221% i forhold til Inverse og 55336% i forhold til Crame.

4.FLERE FLOPS

4.1

A inner product between two vectors u, v ϵ R^n is defined as:

$$\mathrm{u} \cdot \mathrm{v} = \sum_{t=1}^{n} u_t \cdot v_t = u^T \cdot v$$

which can be re-write as:

$$(u_1 \cdot v_1) + (u_2 \cdot v_2) + (u_3 \cdot v_3) + (u_4 \cdot v_4) + \cdots (u_n \cdot v_n)$$

The above expression draws a pattern that it has n times multiplication and n-1 times summation. Therefore, the total flops of the inner product is 2n-1.

4.2

Let x ϵ R^n, so 2-norm of x defined as:

$$\|x\| = \sqrt{{x_1}^2 + {x_2}^2 \ldots + {x_n}^2}$$

The operation of 2-norm of x is actually the square root of inner product of $\mathrm{x} \cdot \mathrm{x}$. The inner product $\mathrm{u} \cdot \mathrm{v}$ has 2n-1 flops:

$$(u_1 \cdot v_1) + (u_2 \cdot v_2) + (u_3 \cdot v_3) + (u_4 \cdot v_4) + \cdots (u_n \cdot v_n)$$

which is same as the part under the square root

$$(x_1 \cdot x_1) + (x_2 \cdot x_2) + (x_3 \cdot x_3) + (x_4 \cdot x_4) + \cdots (x_n \cdot x_n).$$

Moreover, the square root counts one more flop.

Therefor, $\|x\|$ has totally 2n-1+1=2n flops.

4.3

How many flops does the following algorithm use?

```
1 for j = 1 : n
2 for k = 1 : j-1
3 A(:,j) = A(:,j) - ( A(:,j)'*A(:,k) )*A(:,k);
4 end
5 A(:,j) = A(:,j)/norm(A(:,j));
6 end
```

Line 3 has an inner product that counts for 2n-1 flops.

The inner product makes a scalar, then the scalar multiplies a vector with length n, which contributes n more flops. So far the flops is 2n-1+n.

The subtraction of two vectors with length n yields another n flops, now the flops is 2n-1+n+n = 4n-1.

Line 3 is inside the second for-loop with length j-1, so the flops is (4n-1)(j-1).

Since j depends on n, the first for-loop can be counted into (4n-1)(j-1), which yields:

$$\sum_{j=1}^{n}(4\mathrm{n}-1)(\mathrm{j}-1)$$

As n does not involve in the summation, we take 4n-1 out:

$$4\mathrm{n}-1\sum_{j=1}^{n}(\mathrm{j}-1)$$

Can be re-written as:

$$4\mathrm{n}-1\sum_{j=1}^{n}(\mathrm{j})-\mathrm{n}$$

We know that the continuous summation:

$$\sum_{j=1}^{n}(\mathrm{j})=\frac{n(n+1)}{2}$$

So :

$$4\mathrm{n}-1\sum_{j=1}^{n}(\mathrm{j})-\mathrm{n}=(4n-1)(\frac{n(n+1)}{2}-n)$$

Line 5 has a norm operation that counts for 2n flops.

Then the norm divided by a vector with n length, which means n time division. So now we have 2n+n = 3n flops.

Since line 5 is inside the first for-loop that has n length, so the flops is $3n*n = 3n^2$.

Therefore, the total flops for the whole algorithm is:

$$(4n-1)\left(\frac{n(n+1)}{2}-n\right)+\ 3n^2=2n^3+\frac{1}{2}n^2+\frac{1}{2}n$$

The dominated part with highest power is $2n^3$.

5. SPECIALISERET GAUSS-ELIMINATION

5.1

Taking advantage of Hessenberg-matrix could reduce the code to only one for-loop instead of two, as we only need to make the under bi-diagonal 0, which give the same result as conducting gauss elimination. The modified code is underneath:

```
1. A = hess(rand(5));
2. b = rand(5,1);
3. Aug = [A b]; n = size(A,1); nb = n+1;
4. for k = 1 : n-1
5. factor = Aug(k+1,k)/Aug(k,k);
6. Aug(k+1,k) = 0;
7. Aug(k+1,k+1:nb) = Aug(k+1,k+1:nb) - factor*Aug(k,k+1:nb);
8. end
```

5.2

In order to work out the amount of flops in the code, it always starts counting from inside the for-loop. Line 7 has a subtraction and a multiplication. The columns in "Aug(k+1,k+1:nb)" range from k+1 to nb, thus the length of the columns is nb-(k+1)+1. Therefor the total flops for both subtraction and multiplication is 2(nb-(k+1)+1).

Line 6 doesn't involve any (+,-,*,/) operation, so it has 0 flop.

Line 5 has only one division operation.

We have now 2(nb-(k+1)+1)+1 flops inside the for-loop, and $\sum_{k=1}^{n-1} 2(\mathrm{nb} - (\mathrm{k}+1) + 1) + 1$ in total.

Reduction: Nb=n+1

$\sum_{k=1}^{n-1} 2(\mathrm{n}+1-(\mathrm{k}+1)+1)+1$

Take 2 in front:

$2\sum_{k=1}^{n-1}(\mathrm{n}+1-\mathrm{k}-1+1+1/2)$

Reducing:

$2\sum_{k=1}^{n-1}(\mathrm{n}-\mathrm{k}+3/2)$

Decomposition:

$2\sum_{k=1}^{n-1}(\mathrm{n}-\mathrm{k}) + 2\sum_{k=1}^{n-1}(3/2)$

On the left side:

$2\sum_{k=1}^{n-1}(\mathrm{n}-\mathrm{k}) = 2((\mathrm{n}-1)+(\mathrm{n}-2)+(\mathrm{n}-3)+\cdots(\mathrm{n}-\mathrm{n}+1)) = 2(1/2((n-1)^2+(\mathrm{n}-1))) = n^2-n$

On the right side:

$2\sum_{k=1}^{n-1}(3/2) = 2*3/2\sum_{k=1}^{n-1}1 = 3*(\mathrm{n}-1)$

Add left and right side together:

$$n^2 - n + 3 * (\mathrm{n} - 1) = n^2 + 2n - 3$$

So the total flops is $n^2 + 2n - 3$.

6. FEJVURDERING

6.1

The relative error on the right is:

$\frac{\|\tilde{b}-b\|_2}{\|b\|_2} = 0.0116$

The relative error on the left is:

$\frac{\|\tilde{x}-x\|_2}{\|x\|_2} = 0.0227$

6.2

The condition number is:

$$cond[A] = \|A\| \cdot \|A^{-1}\| = 3.9080$$

and the worst case relative error is :

$$cond[A]\frac{\|\tilde{b}-b\|_2}{\|b\|_2} = 0.0453$$

7. BRUG AF FAKTORISERING TIL BILLEDRESTAURERING

7.1

Faktorisering kan gøres på 3 forskellige måder, nemlig Gauss-Elimination, LU-faktorisering og Cholesky.

Den meste håndgribelig metode er Cholesky. Denne metode indeholder 4 krav der skal opfyldes inden den kan benyttes.

1. Symmetrisk
2. Kvadratisk
3. Positiv definit

Som det kan ses matricen opfylder alle betingelser derfor kan man opstille Cholesky faktorisering som er givet ved:

$$A = U^T \cdot U$$

Ved at bruge ligning som er givet i opgaven og erstattes udtrukket for A, fås:

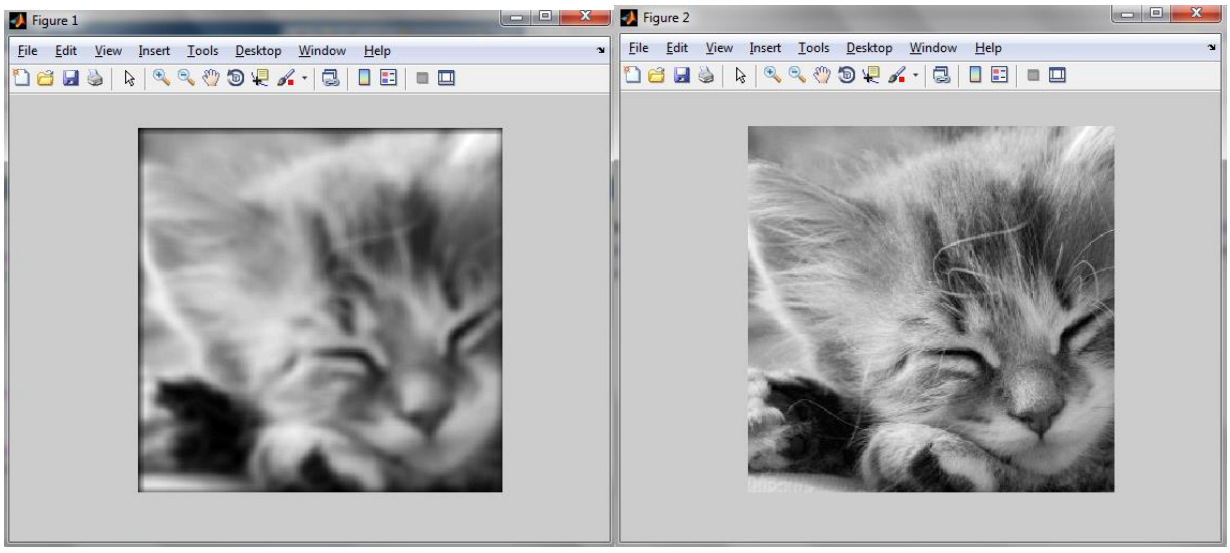
$$B = A \cdot X \cdot A \rightarrow B = U^T \cdot U \cdot X \cdot U^T \cdot U$$

Ved at lave de simple matematiske beregninger kan X findes ved:

$$B \setminus U^T = U \cdot X \cdot U^T \cdot U \rightarrow U \setminus B \setminus U^T = X \cdot U^T \cdot U$$

$$\boxed{U \setminus (U^T \setminus B)/U/\, U^T = X}$$

Det skal bemærkes at $U^T og\ U$ er kendt. U kan nemlig findes fra matlab koden U=chol(A). Dermed faktoriseres kun 1 gang. Ved at sammenligne resultater, det kan nu ses at $X = A \setminus B/A$ er det det samme som $X = U \setminus (U^T \setminus B)$. Det ses at billedet er blevet tydeligere.



Figur 2

```
clear all
clc
load ('deblur.mat')
figure();
imagesc(B), axis image off, colormap gray
%7.1 X = A\B/A; chol funktiion bruges til factorisering (regnes på
%halvdelen da den er symetsik)
U=chol(A);
X = U\(U'\B)/U/U';
x=U\(U'\B);
figure();
imagesc(X), axis image off, colormap gray
figure();
x1 = A\B/A;
imagesc(x1), axis image off, colormap gray
```

Figur 3: Matlab koden for opgave 7.1

7.2

E kan findes ude fra den given ligning i opgaven:

$$\frac{\|\tilde{X} - X\|f}{\|X\|} < Cond[A]^2 \frac{\|E\|_f}{\|B\|_f} < \epsilon$$

Ved at lave isolering fås$\|E\|_f$:

$$\boxed{\|E\|_f < \frac{\epsilon \cdot \|B\|_f}{Cond[A]^2} = \frac{0{,}02 \cdot 1{,}5562 \cdot 10^8}{31{,}3276^2} = 3{,}171242345634412e + 03}$$

7.3

For at finde constant, benyttes samme metode som 7.2:

$$constant \cdot \left\|E_{ny}\right\|_f < \frac{\epsilon \cdot \|B\|_f}{Cond[A]^2} \rightarrow \alpha < \frac{0{,}02 \cdot 1{,}5562 \cdot 10^8}{31{,}3276^2 \cdot \|E\|_f}$$

Næste skridt er at beregne på Xthilde

$$\tilde{X} = U\backslash(U^T\backslash B_{noisy})/U/U^T$$

Hvor B_{noisy}er givet i opgaven som B+E dermed $B_{noisy} = constant \cdot \left\|E_{ny}\right\|_f$

Den relative fejl skal være mindre end 0,02.

$$\boxed{\frac{\left\|\tilde{X} - X\right\|_f}{\|X\|_f} = 1{,}281757324221284e - 04 < 0{,}02}$$

8. LINEÆR DATA-FITTING

8.1

Ved at studere lidt nærmere i ligning 15.11 i bogen og da y og a er defineret som:

$$\bar{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad \bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{50} \end{bmatrix}$$

Ligningssystem struktur kan efterfølges:

$$\bar{\bar{Z}}\bar{a} = \bar{y}$$

Og der med kan Z skrives i en vektor med 3 søjler og 50 rækker

$$\bar{\bar{Z}} = \begin{bmatrix} e^{-\lambda_1 t_1} & e^{-\lambda_2 t_1} & 1 \\ e^{-\lambda_1 t_2} & e^{-\lambda_2 t_2} & 1 \\ \vdots & \vdots & \vdots \\ e^{-\lambda_1 t_{50}} & e^{-\lambda_2 t_{50}} & 1 \end{bmatrix}$$

8.2

Ved at bruge samme metode som er i bogen side 369 opstilles en ligning for a:

$$a = (Z' * Z) \setminus (Z' * y)$$

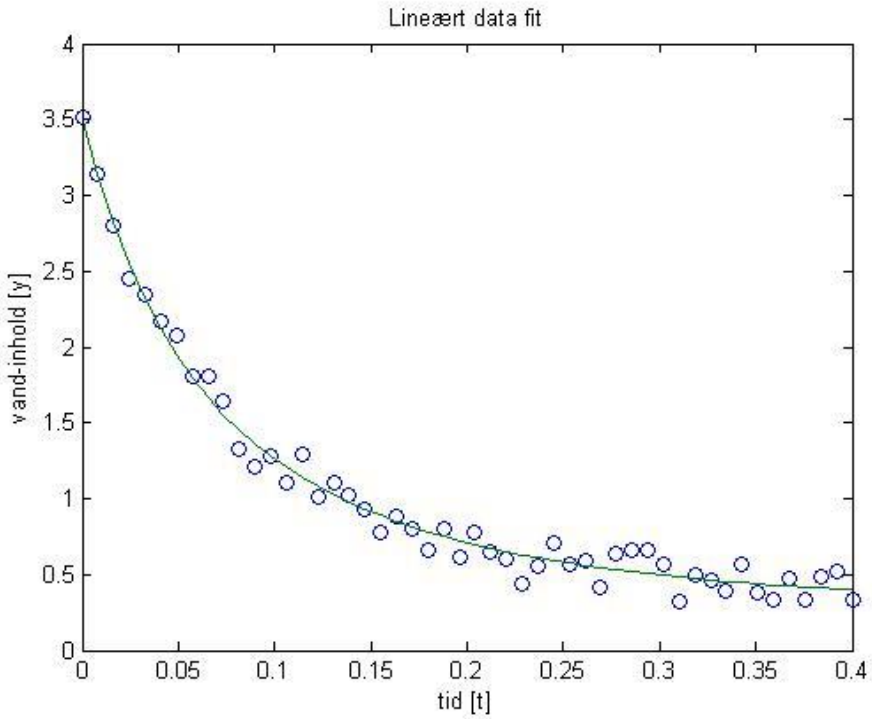
Som ses fra ligning, a findes ved at benytte backslash metoden.

```
%8.2 ligning(15.10
Z=[exp(-lambda1*t) exp(-lambda2*t) ones(50,1)];
%a=Z\y
A = (Z'*Z)\(Z'*y);
%a1=1.2949 a2= 1.8967  a3=0.3222
```

$$\boxed{\bar{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1{,}2949 \\ 1{,}8967 \\ 0{,}3222 \end{bmatrix}}$$

8.3

De fundne værdier for a1, a2 og a3 benyttes til at plotte fit-funktion ϕ(t) med datapunkterne.



Figur 3: Lineært data fiting

Som det ses på figuren, er der opnået meget fint fit.

8.4

Ved at benytte MATLAB indbyggede funktion, findes den største afgivelsen, samt dens placering.

$$f = |\phi(t_i) - y_i|$$

MATLAB->

```
[afgivelse,afpunkt]=max(abs(phi-y));
```

Resultater vil dermed være:

$$maxvalu = 0{,}195564479373723 \qquad i_{maxvalu} = 29$$

9. ULINEÆR DATA-FITTING

9.1

I denne opgave skal der skrives en funktion der hedder fitfun, som skal beregne summen af den kvadrerede afvigelse. Funktion skal have inputtene t,y fra de målte data og så en vektor med værdierne $a_1, \lambda \, og \, a_3$.

Når funktionen får værdierne ind skal den først finde ϕ, hvorefter den skal finde den kvadrerede afvigelse og det giver følgende funktion:

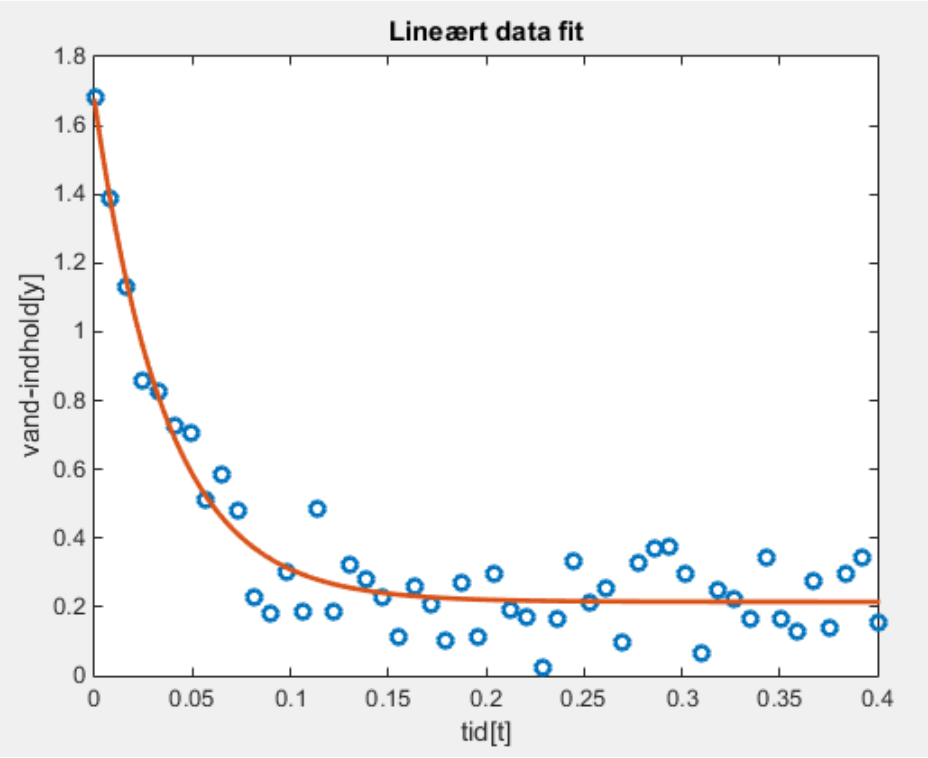
```
1   function [ f ] = fitfun(t,y,p)
2   %Denne funktion først phi funktionen med de inputtede værdier
3   phi=p(1)*exp(-p(2)*t)+p(3);
4   %Derefter bedregners fejlen i anden
5   f=sum((y-phi).^2);
6   end
```

9.2

Til at beregne den mindste λ skal funktionen fminsearch benyttes. Dette gøres ved at opskrive en funktion f som er funktion af p i fitfun. Derefter benyttes fminsearch til at finde en værdi for λ.

Dette giver en $\lambda = 27{,}3324$ og derudover bestemte funktionen også:

$$\boxed{a_1 = 1{,}4557 \; og \; a_3 = 0{,}215}$$



Figur 4:linært data fitting

OPGAVE A. STREGKODELÆSEREN

A.1

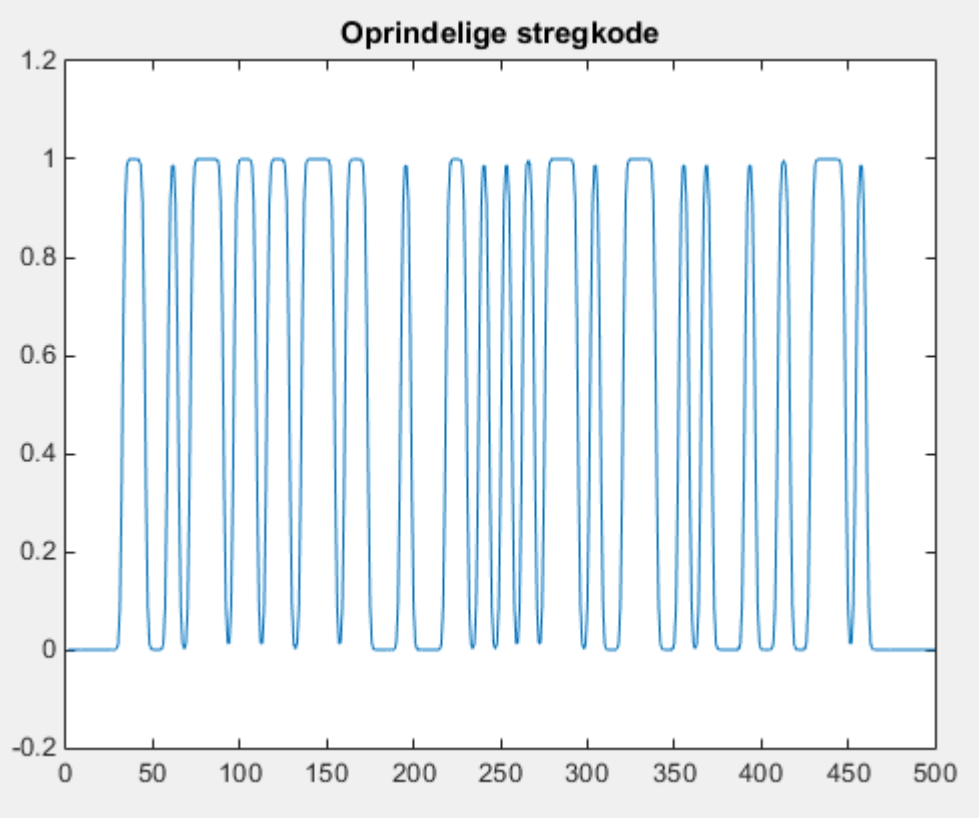
For at beregne et forbedret scan af stregkoden skal korrigerings matricen først opskrives hvilket gøres med formlen:

$$a_{ij} = \begin{cases} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(i-j)^2}{2\sigma^2}\right), & |i-j| \leq 10 \\ 0, & \text{ellers} \end{cases}, \qquad i,j = 1,2,\ldots,n$$

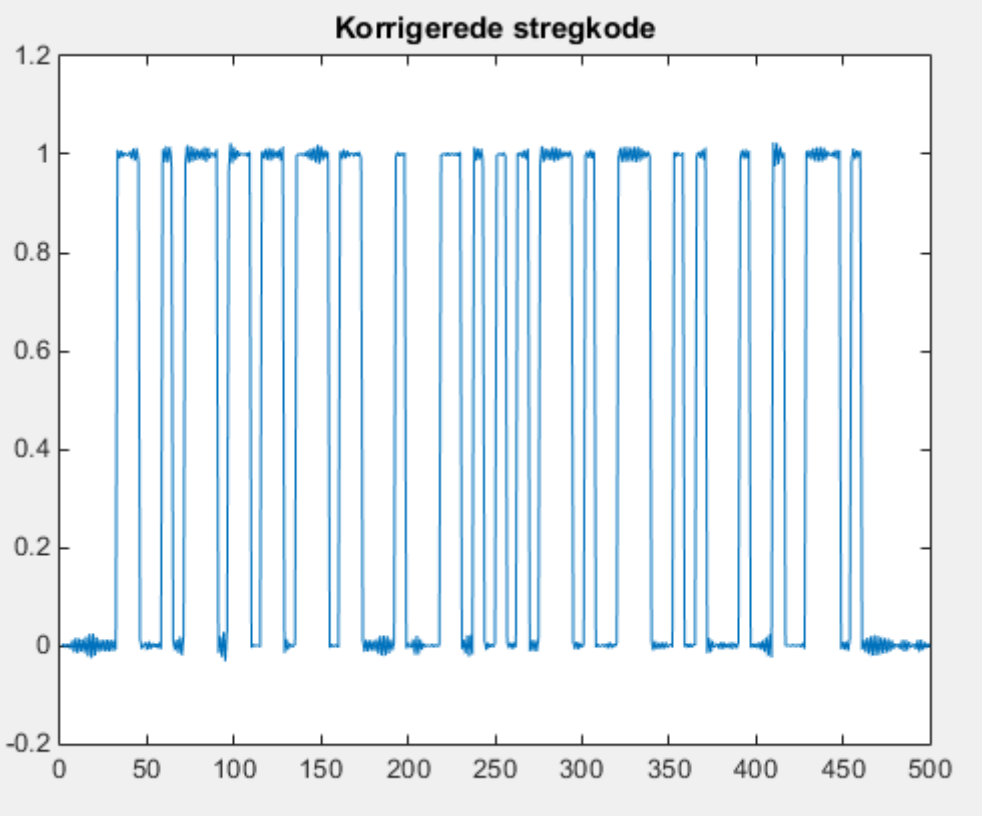
Dette gøres ved at opskrive en for funktion som så gennemregner formlen herover for n=500 gange.

Derefter kan Xtilde findes ved at benytte backslashfunktionen.

Dette giver de følgende to plot.



Figur 5: Oprindelig stregkode



Figur 6: Korrigerede stregkode

A.2

Ved at benytte at $\frac{\left|\left|\tilde{b}-b\right|\right|_2}{\left|\left|b\right|\right|_2} = 1{,}5 * 10^{-4}$ og vi kender

$$\frac{\|\tilde{x}-x\|_2}{\|x\|_2} \leq \mathrm{Cond}[A]\,\frac{\|\tilde{b}-b\|_2}{\|b\|_2}$$

Derved kan vi benytte at ved at beregne cond(a) siden at den beregner den maksimale fejl der må opnås findes ved at lave beregningen:

$$\boxed{cond(A) * \frac{\left|\left|\tilde{b}-b\right|\right|_2}{\left|\left|b\right|\right|_2} = cond(A) * 1{,}5 * 10^{-4} = 0{,}0512}$$

Derved må den maksimale fejl max blive 0,0512

A.3

Den fundende maksimal fejl i delopgave 2 er en del for høj end den ønsket maksimalfejl på 0,01 og derved skal den maksimale relative fejl på dataene findes dette gøres ved:

$$\boxed{\frac{0{,}01}{cond(A)} = 2{,}93 * 10^{-5}}$$

A.4

Først kigges der på forward eliminationen hvor der tilføjes en værdi J som enten er k+10 eller hvis k+10 er højere end n vælger den så bare n. Dette er gør at matricen ikke bliver større end nxn.

Dette benyttes i ligningen $A(i,k+1;n) = A(i,k+1;n) - factor * A(i,k+1;n)$; hvor alle n'erne bliver erstattet af J. Dette gør at der kun beregnes på de 10 felter under diagonalen som der er værdier i som ikke er 0.

```
17      % forward elimination
18 -    for k = 1:n-1
19 -        j=min(k+10,n);
20 -      for i = k+1:j
21 -        factor = A(i,k)/A(k,k);
22 -        A(i,k) = 0;
23 -        A(i,k+1:j) = A(i,k+1:j)-factor*A(k,k+1:j);
24 -        b(i) = b(i)-factor*b(k);
25 -      end
26 -    end
```

I back substitution er der lavet en if statement som gør at der kun regnes på værdier som ikke er nuller.

```
28      % back substitution
29 -    x = zeros(n,1);
30 -    x(n) = b(n)/A(n,n);
31 -    for i = n-1:-1:1
32 -        if i>=n-10
33 -            x(i) = (b(i)-A(i,i+1:1:n)*x(i+1:1:n))/A(i,i);
34 -        else
35 -            x(i) = (b(i)-A(i,i+1:1:i+10)*x(i+1:1:i+10))/A(i,i);
36 -        end
37 -    end
```

Der er lavet en beregnings tidstest på den originale funktion og derefter den modificeret funktion hvilket ændrede sig fra at tage 3 sekunder til at tage 0,064 sekunder hvilket er en godt forbedring.

A.5

Mængden af flops der skal bruges til at lave beregningen har vi bestemt ved først at kigge på forward elimination og derefter back substitution.

I forward elimination kigges der først på den inderste for funktion. Beregningen af faktor bruger et flop og til beregningen af b benyttes der 2 flops da der indgår både en subtraktion og en multiplikation. Den sidste udregning er indeholder 2 flops ved at der er en subtraktion og en multiplikation men der beregnes også på elementer fra k+1 til J så derfor er der her 10 beregninger. Dette giver at A matricen kræver 20 flops og derved bruges der i den inderste for funktion 23 flops og selve for funktionen gennemgås 10 gange for hvor k værdi. Dette giver 230 flops. Den yderste for funktion gennemgås n-1 gange så det totale antal flops til at lave forward elimination er: 230*(n-1) flops hvilket kan antages til at være 230*n flops.

Til back substitution startes der også beregningerne inde i for funktionen. I if funktionen bruges der bruges der 3*n flops og i else funktionen 3*10 flops. Dette giver for if funktionen bruger så 3n+30 og for funktionen gennem køres n gange. Dette giver for back substitutionen $3n^2+30n$

Det totale antal flops nødvendigt til at beregne gauss-funktionen er derfor 230*n flops $+3n^2+30n$ hvilket giver $3n^2+$ 260*n flops.

B. KALIBRERING AF STREGKODELÆSEREN

B.1

25 datapunkt (y1, y2….y25) skal fittes med nedstående formel:

$$b_i = A_{i,10} = \frac{1}{\sigma \cdot \sqrt{2\pi}} exp\left(-\frac{i-10}{2\sigma^2}\right)^2$$

Den formel kan opskrives i MATLAB:

```
for i=1:25
    if abs(i-1)<=10;
        b(i)=(1/(sigma*sqrt(2*pi)))* exp(-((i-10)^2)./2*(sigma).^2);
    else
       b(i)= 0;
    end
end
```

Ved at bruge en (for-else) funktion, kan opstilles de betingelser der er afsat i opgaven.

B.2

I den første skridt, skal den summere residual fejl findes ved at opstilles en funktion som det nu kaldes ”fitfunb” og er defineret som:

$$f = \sum_{i=1}^{n} (\phi(t_i) - y_i)^2$$

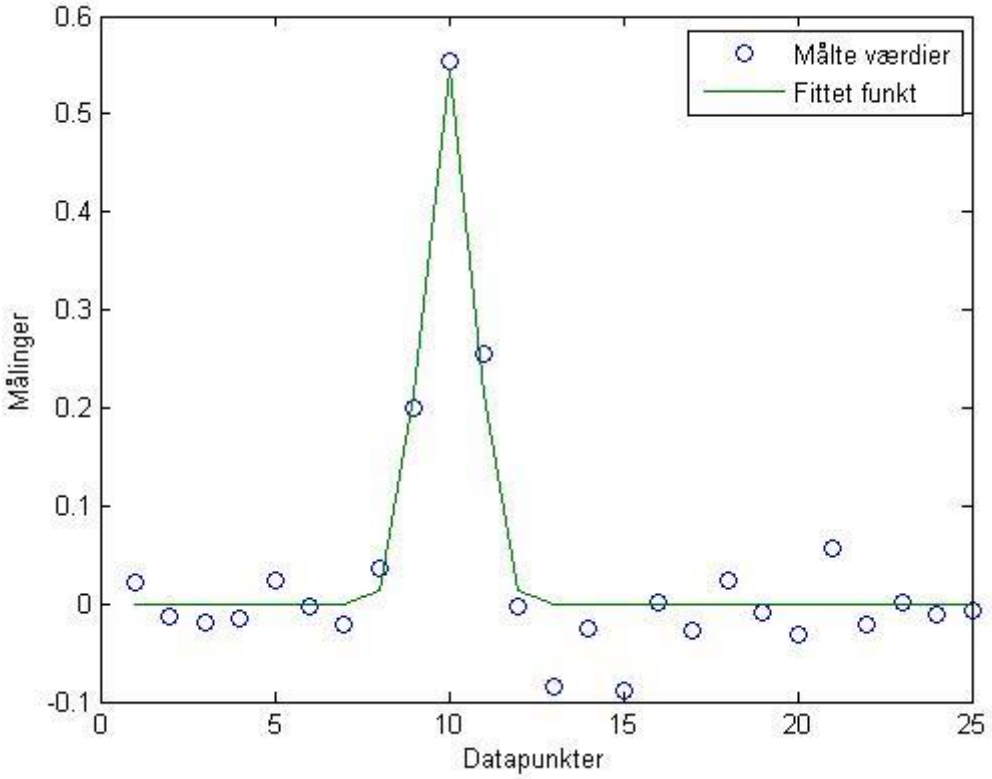
```
function err = fitfunb(I,Y,sigma)
A=(1/(sigma*sqrt(2*pi)))*exp(-((I-10).^2)/(2*(sigma)^2))

err = sum((Y-A).^2);
end
```

Dermed optimeres denne, vha. ”fminbnd” som er givet i opgaven med to grænser. Ved at udføre disse beregninger i MATLAB, fås:

$$\boxed{\sigma = 0.732164388460562}$$

Figuren viser plottet af Funktion $\phi(t)$ med de fundne datapunkter og parameter.



Figur 7

C. FROBENIUS-NORMEN AF EN MATRIX

The Frobenius norm is sometimes called the Euclidean norm, A is a n x n matrix, the Frobenius norm of a matrix $\|A\|_f$ can be expressed as:

$$\|A\|_f = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}$$

We can see that it is defined as the square root of the sum of the absolute squares of each elements.

Since it has n x n elements, the absolute squares of each elements makes n^2 flops.

The sum of each elements contributes $n^2 - 1$ flops.

The square root adds one more flop.

So the total flops of the Frobenius norm is $n^2 + n^2 - 1 + 1 = 2n^2$

TEORI-SPØRGSMÅL

I: LU-FAKTORISERING

LU-faktorisering er at opdele A matricen i en upper og lower matrice. Den upper matrice svarer til den øverste halvdel af Gauss-eliminationen og den lower matrice er en matrice med 1 i diagonalen og derunder er eliminations koeffienterne. Eliminations koefficienterne er de faktorer som skal benyttes for at få nul på pladserne under diagonalen i Gauss-eleminationen.

LU-faktorisering bruges i Gauss-eleminationen da der laves nuller under diagonalen vil man derfor finde den upper matrice. De faktorer som så skal ganges på b-matricen bliver i stedet gemt i en ny matrice (lower).

Det vides så at **LU=A** men til at finde **x** i det nye system skal en erstatning for b matricen så findes. Dette gøres ved at lave beregningen **Ld=b** hvor man så kan finde **x** ved at lave beregningen i **Ux=d**.

II: BEVIS FOR LU-FAKTORISERING

II.1

I denne opgave skal vise at der laves en række operation på række nr. 2 ved at benytte en L-matrice

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} og\ l_{2.1} = \begin{bmatrix} 1 & 0 & 0 \\ x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Så kan produktet af disse to matricer så findes:

$$l_{2,1} * A = \begin{bmatrix} a & b & c \\ a*x+d & b*x+e & c*x+f \\ g & h & i \end{bmatrix}$$

Her kan ses at denne operation laver en række operation på anden række med værdierne fra den første række.

II.2

$l_{3,1}$ laver operationer på 3. række ud fra værdierne i den første række og $l_{3,2}$ laver operationerne på den 3. række ud fra værdierne i 2. række.

$$l_{3,1} * A = \begin{bmatrix} a & b & c \\ d & e & f \\ a*x+g & b*x+h & c*x+i \end{bmatrix}$$

$$l_{3,2} * A = \begin{bmatrix} a & b & c \\ d & e & f \\ d*x+g & e*x+h & f*x+i \end{bmatrix}$$

II.3

Gauss-eliminationens formål er at lave nuller under diagonalen. Så $l_{2,1}(x_1)$'s opgave er at værdien i $A_{2,1} = 0$ derfor skal den så finde en passende x_1-værdig til denne beregning. Liggende skal $l_{3,1}(x_2)$ lave $A_{3,1} = 0$ og $l_{3,2}(x_3)$ skal sørge for at værdien $A_{3,2} = 0$.

Som eksempel til at finde x_1 laves beregningen: $a * x_1 + d = 0 \rightarrow a * x_1 = -d \rightarrow x_1 = -\frac{d}{a}$ og liggende beregning gøres for de andre og der findes så $x_2 = -\frac{g}{a}\ og\ x_3 = -\frac{e}{h}$

II.4

Ud fra at vide at L*A=U så må de forskellige l værdier være den inverse værdig af dem som er i L matrixen.

$$l_{2,1}(-x_1) * l_{3,1}(-x_2) * l_{3,2}(-x_3) = \begin{bmatrix} 1 & 0 & 0 \\ -x_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_2 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -x_3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -x_1 & 1 & 0 \\ -x_2 & -x_3 & 1 \end{bmatrix}$$

III: CHOLESKY-FAKTORISERING

LU-faktorisering kræver kun at matricen er kvadratisk hvor Cholesky-faktoriseringen kræver mere af matricen. Cholesky kræver at den er kvadratisk, symmetrisk og positiv definit. Grunden til at man vælger Cholesky-faktorisering er at den kræver mindre arbejde og er derved hurtigere end LU-faktoriseringen da man benytter at matricen er symmetrisk.

IV: ØVRE GRÆNSE FOR RELATIV FEJL

Det betragtes at de to ligninger $Ax = b$ og $Ax' = b'$ kan omskrives til : $Ax - b = 0$ og $Ax' - b' = 0$ dermed kan de skrives som:

$$Ax - b = A\tilde{x} - \tilde{b} \rightarrow x - \tilde{x} = A^{-1}(b - \tilde{b})$$

For den anden ligning kan forholdet skrives som:

$$\Delta x = x - \tilde{x} \rightarrow \|\Delta x\|_2 = \|\tilde{x} - x\|_2$$

Ved at benytte den først ligning:

$$\|x - \tilde{x}\| = \left\|A^{-1} * \left(\tilde{b} - b\right)\right\|_2$$

Ved at benytte kanonformlen:

$$\|Ax\|_2 \leq \|A\|_2 \|x\|_2$$

Dermed fås:

$$\|x - \tilde{x}\| \leq \|A^{-1}\|_2 \left\|\left(\tilde{b} - b\right)\right\|_2$$

Ligning i opgaveteksten kan fås:

$$Ax = b \rightarrow \|Ax\|_2 = \|b\|_2$$

$$\|b\|_2 \leq \|A\|_2 \|x\|_2$$

Ved at gøre brug af denne ligning og ombytte fås:

$$\|A\|_2 \geq \frac{\|b\|_2}{\|x\|_2}$$

Indsættelse i den første afmærkeret ligning:

$$\|\tilde{x} - x\|_2 = \frac{\|\tilde{x}\|_2}{\|b\|_2} \|(\tilde{b} - b)\|_2$$

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leq \|A\|_2 \cdot \|A^{-1}\|_2 \frac{\|\tilde{b} - b\|_2}{\|b\|_2}$$

$$Cond[A] = \|A\|_2 \cdot \|A^{-1}\|_2 \rightarrow \frac{\|\tilde{x}-x\|_2}{\|x\|_2} \leq Cond[A] \frac{\|\tilde{b}-b\|_2}{\|b\|_2}$$

Hvor udtrykket for den relative fejl er bevist.

V: HVAD GÅR DATAFITTING UD PÅ?

Når data fittes findes en funktion $\phi(t_i)$ der beskriver dataene bedst muligt. Afvigelsen fra data beskrives som summen af residual fejlene: $\sum|\phi(t_i) - y_i|$. for i=1,2,..n. Ved datafitting ønskes at minimere fejlen mellem datasæts målingerne og en tilnærmet funktion ($\phi(t_i)$).

VI: LINEÆR OG ULINEÆR DATAFITTING

Lineær datafitting: funktionen $\phi(a; x)$ knytter sig til parametrene ai og de kan findes ved normalligningerne.

Ulineær datafitting: mindst én af parametrene ai I funktionen $\phi(a; x)$ optræder ulineært, dette kan ikke løses via sammenhængen $\bar{\bar{Z}}\bar{a} = \bar{y} \leftrightarrow \bar{a} = \bar{\bar{Z}}\backslash\bar{y}$. Ulineær kan løses ved at omskrive det ulineær udtryk til lineært (f.eks. log funktion) og hvis dette ikke kan lade sig gøre, løses ved at finde minimummet af de kvadrede residualer.

APPENDIX – MATLAB KODE

OPGAVE 2

```
clear all; clc;
%opgave 2.1
%Matricen A med mængderne og vektoren B med totale masser opskrives
A=[3 12 10; 12 0 20;0 2 30];
B=[72.3 ; 99.5; 56.6];
%opgave 2.2
%Masserne af søm, skruer og møtrikker findes.
x=round(A\B,2);
%opgave 2.3
%den nye total masse matrice opskrives
Bny=[73.3;98.4;57.1];
%Masserne af søm, skruer og møtrikker findes.
xny=round(A\Bny,2);
%opgave 2.4
%Differencen på de to funden værdier findes
diff=abs(x-xny);
[maxfejl maxindex]=max(diff);
```

OPGAVE 3 .2

```
clear all
close all
clc
%det er lavet en funktion hvilken skal vælges en n for at kunne beregne .
[t_100,backlash_100,inverse_100,x_100]=beregningstider(100);
sort(t_100);
%alle de 3 metoder skal give samme resultat ved at vælge n=tal

%3.2 x repræsenter cram ogt for tiden
[t_100,backslash_100,inverse_100,x_100]=beregningstider(100);
[t_200,backslash_200,inverse_200,x_200]=beregningstider(200);
[t_400,backslash_400,inverse_400,x_400]=beregningstider(400);
[t_800,backslash_800,inverse_800,x_800]=beregningstider(800);
A = [t_100;t_200;t_400;t_800];
T = array2table(A,'VariableName',{'backlash' 'inverse' 'Crame'});
save T
save T.txt A -ascii
%det kommer ud som en T.txt fil som backlash inverse og cramer i 3 søjler
```

OPGAVE 7.2 OG 7.3

```
%7.2
Ef=0.02*norm(B,'fro')/cond(A)^2;
%svaret er 3.171242345634412e+03

%7.3
[n,n]=size(A);
E=rand(n);
syms k
scale=solve(norm(E,'fro')*k==Ef,k);
E=E*eval(scale);
Bn=B+E;
Xt=U\(U'\Bn);
fejl=norm (Xt-x,'fro')/norm(x,'fro');
%relative fejl er på 1.2907e-04 som er mindre end 0.02
```

OPGAVE 8

```
clear all
clc
load ('NMRdata8.mat')
lambda1=27;
lambda2=8;
%8.1 Det laves en matrice Z der består delen i phi matricen i opgaveteskten
%(ligning(15.11) bogen

%8.2 ligning(15.10
Z=[exp(-lambda1*t) exp(-lambda2*t) ones(50,1)];
%a=Z\y
A = (Z'*Z)\(Z'*y);
%a1=1.2949 a2= 1.8967  a3=0.3222

%8.3
a1=1.2949; a2= 1.8967;  a3=0.3222;
phi=a1*exp(-lambda1*t)+a2*exp(-lambda2*t)+a3;
plot(t,y,'o',t,phi);
title ('Lineært data fit'); xlabel('tid [t]');ylabel('vand-inhold [y]')

%8.4
[afgivelse,afpunkt]=max(abs(phi-y));
%afgivelse=0.1956 og afgivelse_punkt=29
```

OPGAVE 9 -FITFUN

```
function [ f ] = fitfun(t,y,p)
%Denne funktion først phi funktionen med de inputtede værdier
phi=p(1)*exp(-p(2)*t)+p(3);
%Derefter bedregners fejlen i anden
f=sum((y-phi).^2);
end
```

Matlabkode

```
clc; clear all; close all;
load('NMRdata9.mat');
%%
%Opg. 9.2
%Først laves en funktion af p i fitfun
f=@(p) fitfun(t,y,p);
%så benyttes fminsearch funktionen til at finde værdierne til p
a=fminsearch(f,[0.1 0.1 0.1])
%Så opskrives ligningen på ny.
tt = linspace(0,0.4,300)';
yy = a(1)*exp(-a(2)*tt)+a(3);
plot(t,y,'o',tt,yy,'-','linewidth',2);
title('Lineært data fit')
ylabel('vand-indhold[y]')
xlabel('tid[t]')
```

OPGAVE A – MATLAB KODE

Matlab koden

```
clc; clear all; close all;
%A.1
load('barcode.mat');
sigma=1.15;
figure();
plot(btilde)
%A-matricen opskrives
title('Oprindelige stregkode')
for i=1:500
    for j=1:500
        if abs(i-j)<=10
            a(i,j)=1/(sigma*(sqrt(2*pi)))*exp(-(i-j)^2/(2*sigma^2));
        else
            a(i,j)=0;
        end
    end
end
Xtilde=a\btilde;
figure();
plot(Xtilde)
title('Korrigerede stregkode')
%%
%Opg. A.2
%Fejlen findes
b=cond(a)*1.5*10^-4;
%%
%Opg. A.3
maxerr=0.01/cond(a);
%%
%Opg. A.4
%Der tages tid på den oprindelige funktion
tic
x1=GaussNaiveO(a,btilde);
toc
t(1)=toc
%Der tages tid på den optimerede funktion for at se forbedringen
tic
x2=GaussNaive(a,btilde);
toc
t(2)=toc
```

Den optimerede gauss-elemination

```
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end

% forward elimination
for k = 1:n-1
    j=min(k+10,n);
  for i = k+1:j
    factor = A(i,k)/A(k,k);
    A(i,k) = 0;
    A(i,k+1:j) = A(i,k+1:j)-factor*A(k,k+1:j);
    b(i) = b(i)-factor*b(k);
  end
end

% back substitution
x = zeros(n,1);
x(n) = b(n)/A(n,n);
for i = n-1:-1:1
    if i>=n-10
        x(i) = (b(i)-A(i,i+1:1:n)*x(i+1:1:n))/A(i,i);
    else
        x(i) = (b(i)-A(i,i+1:1:i+10)*x(i+1:1:i+10))/A(i,i);
    end
end
```

OPGAVE B

```
clear all
clc
%B1
%startgæt sigma=3
sigma=3
for i=1:25
    if abs(i-1)<=10;
        b(i)=(1/(sigma*sqrt(2*pi)))* exp(-((i-10)^2)./2*(sigma).^2);
    else
       b(i)= 0;
    end
end

%B2
load('kalibreringsdata.mat');
f=@ (sigma) fitfunb(I,Y,sigma);
sigma = fminbnd(f,0,10);
%sigma=0.73
A=(1/(sigma*sqrt(2*pi)))*exp(-((I-10).^2)/(2*(sigma)^2));
plot(I,Y,'O',I,A);
xlabel('Datapunkter'), ylabel('Målinger'), legend('Målte værdier','Fittet funkt')
```