Case 1 Vejecelle

Gruppe 2

Studienummer	Navn	Studieretning
202310755	Peter Thule Kirketerp Linstad	E
202004347	Otto Sejrskild Santesson	E
202001087	Mudar Issam	E

Contents

Indledning og formål 3
Opgave 1 – Data analyse 3
1.1 Middelværdi, spredning og varians 3
Opgave 2 – Design af midlingsfilter 6
2.1 Midlingsfilter design 8
2.2 Histogrammer 8
2.3 Maksimale længde af FIR midlingsfilter 9
2.4 Eksponentielt midlingsfilter design. 10
2.4.1 Eksponentielt midlingsfilter med α = 0,1 11
2.4.2 Eksponentielt midlingsfilter med α = 0,5 12
2.4.3 Eksponentielt midlingsfilter med α = 0,9 13
2.5: 100. ordens FIR midlingsfilter og α-værdien 14
2.6: Korrupt data 15
Opgave 3 - System overvejelser 15
2. Konklusion. 16
3. Referencer. 17

Indledning og formål

I denne øvelse undersøges et målesystem bestående af en vejecelle, som bruges til at registrere vægt. Som i alle målesystemer vil der være en vis støj og usikkerhed på signalet, hvilket kan påvirke nøjagtigheden af målingen og aflæsningen.

Formålet med øvelsen er at analysere og forbedre målesignalet fra en vejecelle ved hjælp af digital signalbehandling i MATLAB. Vi skal kvantificere støj og usikkerhed i det rå signal, designe og implementere forskellige midlingsfiltre (både FIR og eksponentielle), samt vurdere effekten af disse filtre på støjreduktion, indsvingningstid og målepræcision.

Gennem arbejdet får vi en praktisk forståelse af, hvordan man kan optimere et målesystem ved at anvende filtrering og statistisk analyse af signaldata. Resultaterne giver indsigt i forholdet mellem filterorden, støjreduktion og responstid, hvilket er centralt i designet af præcise vægtsystemer.

Opgave 1 – Data analyse

1.1 Middelværdi, spredning og varians

Først har vi delt vejecelle signalet op i hhv. belastet og ubelastet, for at undgå signalets transienter. På denne måde bliver vores beregninger mere retvisende.

Til at finde middelværdi, spredning og varians har vi brugt Matlabs indbyggede funktioner med følgende kode i Figur 1:

```
%% Opg 1 - Data analyse

mean_belastet = mean(x_belastet);
mean_ubelastet = mean(x_ubelastet);

var_belastet = var(x_belastet,1);
var_ubelastet = var(x_ubelastet,1);

std_belastet = std(x_belastet,1);
std_ubelastet = std(x_ubelastet,1);

fprintf('Belastet (1 kg):   mean = %.6g, var = %.6g, std = %.6g\n', mean_belastet, var_belastet, std_belastet);
fprintf('Ubelastet (0 kg):  mean = %.6g, var = %.6g, std = %.6g\n\n', mean_ubelastet, var_ubelastet, std_ubelastet);
```

Figur 1 Kode til beregning af middelværdi, varians og spredning af det opdelte signal

Her får vi nedenstående resultater på Figur 2

```
Belastet (1 kg):   mean = 1107.35, var = 662.666, std = 25.7423
Ubelastet (0 kg):  mean = 1405.79, var = 798.581, std = 28.2592
```

Figur 2 screenshot af command view af beregnede værdier

Disse beregnede værdier vil vi bruge til at sammenligne med histogrammerne for de to opdelte signaler.

1.2 Plots af histogrammer

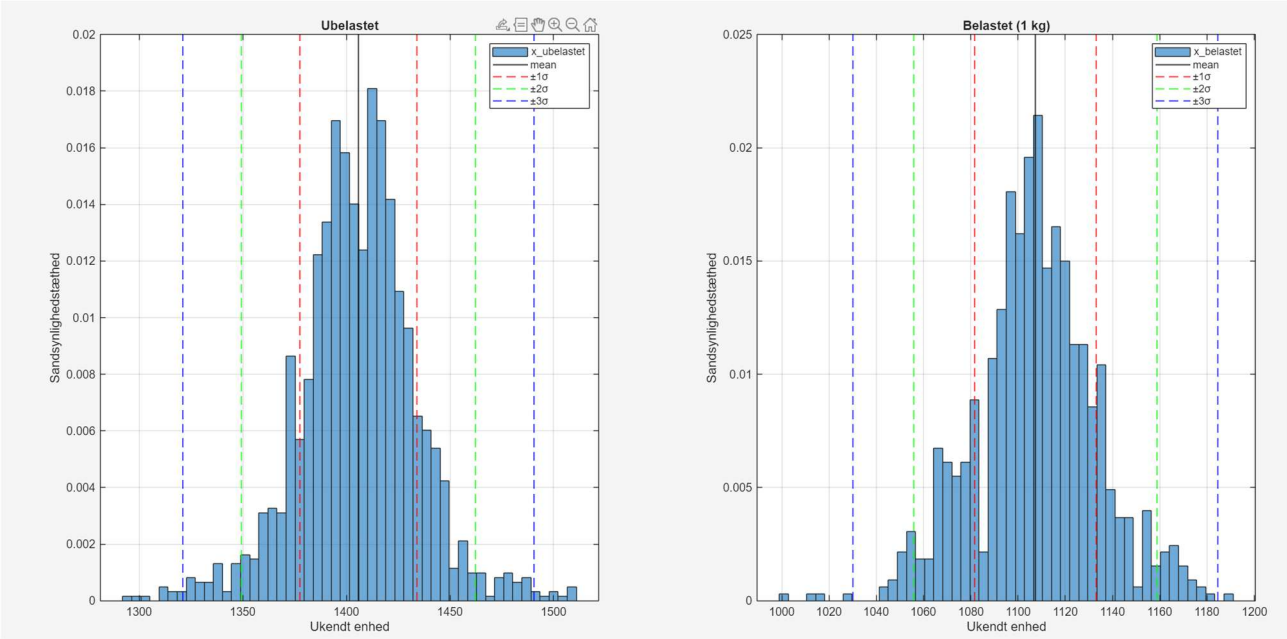
Vi plotter to histogrammer af hhv. det ubelastede og belastede vejecelle signal.

Nedenstående kode viser kodeeksempel på plot af histogram for det ubelastede signal:

```
%% Opg 1 - Histogrammer
% Histogrammer med ±1σ, ±2σ, ±3σ linjer
figure;

% ---------- Ubelastet ----------
subplot(1,2,1)
histogram(x_ubelastet, 'Normalization', 'pdf',NumBins=50); hold on; grid on
title('Ubelastet');
xlabel('Ukendt enhed'); ylabel('Sandsynlighedstæthed');

% Lodrette linjer ved ±1σ, ±2σ, ±3σ
mu = mean_ubelastet;
sigma = std_ubelastet;
xline(mu, 'k-', 'LineWidth', 1.2, 'DisplayName','mean');
xline(mu + sigma, 'r--', 'LineWidth', 1.2, 'DisplayName','±1σ');
xline(mu - sigma, 'r--', 'LineWidth', 1.2, HandleVisibility='off');
xline(mu + 2*sigma, 'g--', 'LineWidth', 1.2, 'DisplayName','±2σ');
xline(mu - 2*sigma, 'g--', 'LineWidth', 1.2, HandleVisibility='off');
xline(mu + 3*sigma, 'b--', 'LineWidth', 1.2, 'DisplayName','±3σ');
xline(mu - 3*sigma, 'b--', 'LineWidth', 1.2, HandleVisibility='off');
legend('show');
```

Figur 3 kodeudsnit af histogram-plot af ubelastet signal.

Med denne kode på Figur 3 får vi følgende histogrammer på Figur 4



Figur 4 Histogram med middelværdi og spredning for hhv. ubelastet og belastet vejecelle signal

Histogrammerne for både den belastede og ubelastede tilstand viser en tydelig klokkeform, hvilket tyder på, at målesignalet (og støjen) er tilnærmelsesvist normalfordelt.

Den målte spredning (standardafvigelse) stemmer også godt overens med histogrammets bredde:

Flertallet af samples ligger indenfor ca. ±1 til ±2 standardafvigelser fra middelværdien, som forventet for en normalfordeling.

Hvis vi helt konkret sammenligner målte værdier for det belastede signal med histogrammet, kan vi se følgende:

- Den målte middelværdi er på 1107, hvilket umiddelbart stemmer fint overens med aflæsning på histogrammet.
- Den målte spredning ligger på 25,7 og ved at aflæse histogrammet får vi (tilnæmelsesvist) en spredning på $1132 - 1107 \cong 25$. Det skal dog bemærkes at værdierne på histogrammet er aflæst ved at kigge på figuren og er derfor et estimat, men det kan dog også ses at den målte og aflæste værdi er meget tæt på hinanden.

1.3 Plot af frekvensspektret

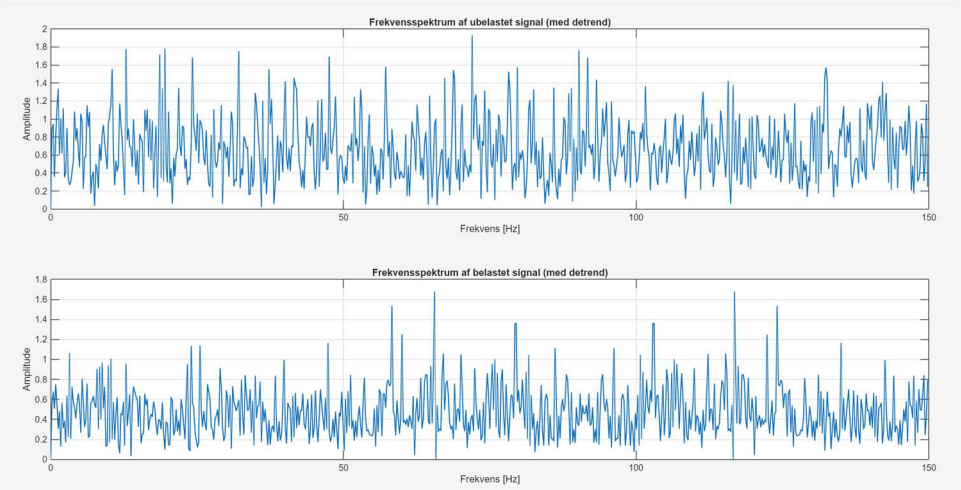
For at plotte frekvensspektret laver vi først FFT (Fast Fourier Transformation) af det opdelte signal. Til dette bruger vi Matlabs indbyggede funktion fft(), som det ses på Figur 5.

Det skal her bemærkes at der benyttes funktionen detrend() til at fjerne DC-offset. Vi forsøgte i starten at fjerne DC-offset, ved blot at trække middelværdien fra signalerne, men fik stadig en markant frekvenskomponent, meget tæt på DC, hvilket kunne tyde på noget drift. Derfor bruges detrend-funktionen, da denne også fjerner lineære trends.

```
% FFT af ubelastet signal – brug detrend til at fjerne offset/trend
Nf = length(x_ubelastet);              % antal samples

% Fjern både DC-offset og evt. langsom drift (lineær trend)
x_fft_u = detrend(x_ubelastet);
x_fft_b = detrend(x_belastet);

% Beregn FFT
X_u = fft(x_fft_u);
X_b = fft(x_fft_b);

% Frekvensakse (0 ... fs)
f = (0:Nf-1)*(fs/Nf);

% Plot kun den positive halvdel af spektret
figure;
subplot(2,1,1);
plot(f(1:floor(Nf/2)), abs(X_u(1:floor(Nf/2)))/Nf, 'LineWidth', 1.2);
xlabel('Frekvens [Hz]');
ylabel('Amplitude');
title('Frekvensspektrum af ubelastet signal (med detrend)');
grid on;
```

Figur 5 Kodeudsnit af FFT af de opdelte signaler samt plot af frekvensspektret for det ubelastede signal

Her får vi følgende frekvensspektre på Figur 6:



Figur 6 Frekvensspektre for hhv ubelastet og belastet vejecelle signal.

Spektret af det detrendede ubelastede signal fremstår bredbåndet og uden dominerende frekvenskomponenter, hvilket er karakteristisk for hvid støj.

Selvom amplituden i spektret ikke er nul, er det netop forventeligt, da støjens energi er fordelt tilfældigt og jævnt over hele frekvensbåndet.

Dette bekræfter, at den pålagte støj i signalet tilnærmelsesvist opfører sig som hvid støj.

1.4 Afstand mellem bit-niveauer i gram

For at kunne bestemme afstanden imellem bit-niveauer, altså værdien af LSB, skal vi bruge gennemsnittet af hhv. det ubelastet og belastede signal. Forskellen mellem disse to målinger angiver, hvor mange counts den kendte vægt svarer til. Værdien af ét count (LSB) beregnes derefter som den kendte vægt divideret med forskellen i counts. Herved fås, hvor mange gram ét digitalt trin svarer til (g/count).

Følgende kode på Figur 7 viser implementeringen af det.

```
delta_counts = mean_ubelastet - mean_belastet;
LSB_g = 1000 / delta_counts;
LSB_kg = 1 / delta_counts;
fprintf('LSB = %.4f g/count (%.6f kg/count)\n\n', LSB_g, LSB_kg);
```

Figur 7 kodeudsnit til beregning af LSB-værdi

Som vi kan se på nedenstående Figur 8 har vi en afstand imellem bit niveauerne på 3,3507 g/count.

```
LSB = 3.3507 g/count (0.003351 kg/count)
```

Figur 8 Screenshot af command view med fprintf af LSB-værdi angivet i g/count og kg/count

Opgave 2 – Design af midlingsfilter

2.1 Implementering

For at implementere midlingsfiltre med orden 10, 50 og 100 tager vi udgangspunkt i Ligning 1:

$$y[n] = \frac{1}{M}\sum_{m=0}^{M-1} x[n-m]$$

Ligning 1 – Midlingsfilterformlen

Vi implementerer dette som en funktion i Matlab kode, så man nemt kan ændre på ordenen af midlingsfilteret:

```
function y = moving_average_filter(x, M)

y = zeros(size(x)); % Preallocate y for efficiency

for n = 1:length(x)
    if n <= M
        y(n) = sum(x(1:n));
    else
        y(n) = sum(x(n-M+1:n));
    end
end
y=y./M;
end
```

Figur 9 – Kodeudsnit af vores midlingsfilterfunktion; x er inputsignalet og M er ordenen på filteret

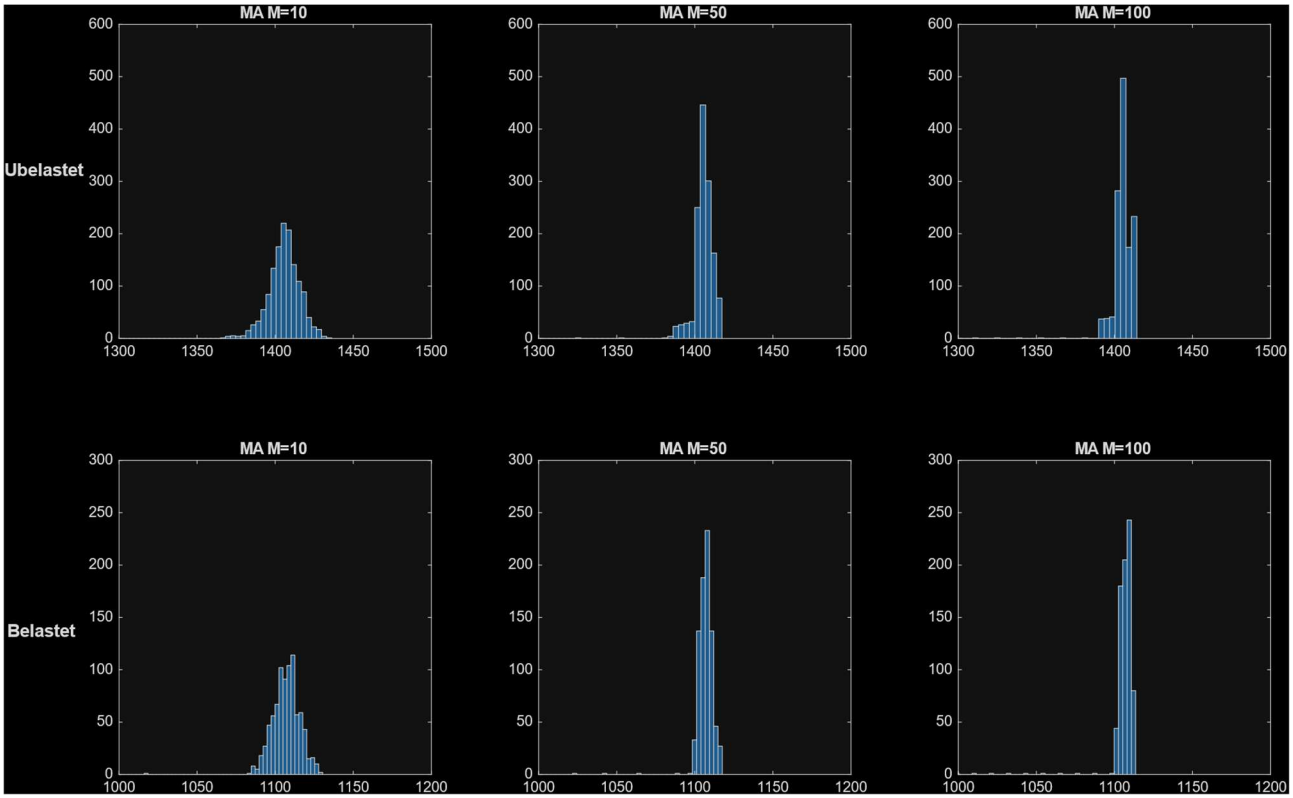
2.2 Histogrammer

Ifølge teorien for støjvarians og midlingsfilter, vil vi kunne forvente følge relation:

$$\sigma_{efter\_filter}{}^2 = \frac{\sigma_{før\_filter}{}^2}{M}$$

Ligning 2 – Formel for variansreduktion givet et midlingsfilter med længde M.

Varians bør derfor falde med en faktor svarende til ordenen for filteret. Kort sagt, bør vi derfor se en mindre spredning jo højere op i filterorden vi går. Det gør vi heldigvis også, når vi plotter spredningen i nedenstående histogrammer (Figur 10).



Figur 10 – Histogrammer af spredningen af signalet for den ubelastede og den belastede del. En større spredning er et udtryk for mere støj. Jo højere filterordenen bliver, jo mindre er spredningen – støjen mindskes derfor.

Vi kan også prøve at se hvordan de konkrete varianser er for signalet efter de forskellige filtreringer; se Figur 8.

```
LSB = 3.3507 g/count (0.003351 kg/count)
MA M=10 | Var belastet: 68.8951 | Var ubelastet: 94.8149
MA M=50 | Var belastet: 13.5867 | Var ubelastet: 28.8512
MA M=100 | Var belastet: 8.5657 | Var ubelastet: 21.4362
```

Figur 11 – Beregnet varianser for de to dele af signalet (belastet og ubelastet) efter filtrering ved forskellige filterordener (10, 50 og 100)

Vi ser, at reduktionen af variansen ikke følger den teoretiske reduktion (se Ligning 2). Dette kan skyldes, at vores sample size er relativt lille, hvilket kan have indflydelse på variansen. Det kan også skyldes at støjen der er på signal ikke er 100 % hvid, men kun tilnærmelsesvis.

2.3 Maksimale længde af FIR midlingsfilter.

I et vejesystem kan man kræve, at indsvingningstiden højst er 100 ms. Det betyder, at systemets respons skal stabilisere sig inden for denne tid. For at sikre dette beregner man den maksimale filterlængde for et FIR-midlingsfilter.

Indsvingningstiden afhænger af filterlængden N, fordi hvert output er gennemsnittet af N inputprøver. Filterlængden findes med formlen:

$$N = \text{Indsvingningstid} \cdot \text{samplingsfrekvensen}$$

$$N = 300Hz \cdot 0.1s = 30$$

hvis maximale sample-tidene er 100ms, så skal vi bruge en filterlængde på 30.

```
%% Beregn den maksimale af FIR-midlingsfilteret:

% Definer den ønskede maksimale indsvingningstid i sekunder
max_indsvingningstid_sec = 0.1; % 100 millisekunder

% Beregn det tilsvarende antal samples baseret på samplingsfrekvensen
max_indsvingningstid_samples = max_indsvingningstid_sec * fs;

% Beregn den maksimale længde af FIR-midlingsfilteret
max_fir_length = floor(max_indsvingningstid_samples);

disp(['Den maksimale længde af FIR-midlingsfilteret: ', num2str(max_fir_length), ...
    ' Samples']);
```

Udprint fra terminalen:

```
Den maksimale længde af FIR-midlingsfilteret: 30 Samples
```

Figur 12 koden til at beregne den maksimale af FIR-midlingsfilteret.

2.4 Eksponentielt midlingsfilter design.

Et eksponentielt midlingsfilter bruges til at glatte signaler. Hver outputværdi beregnes som en vægtet sum af den aktuelle inputværdi og den forrige outputværdi.

Filteret styres af parameteren α, som bestemmer vægten mellem nuværende og tidligere værdier. Typisk gælder: $0 < \alpha < 1$

Vi bruger formlen [2] som vist i Figur til at designe det eksponentielle midlingsfilter.

Figur 13 Eksponentielt midlingsfilter formlen

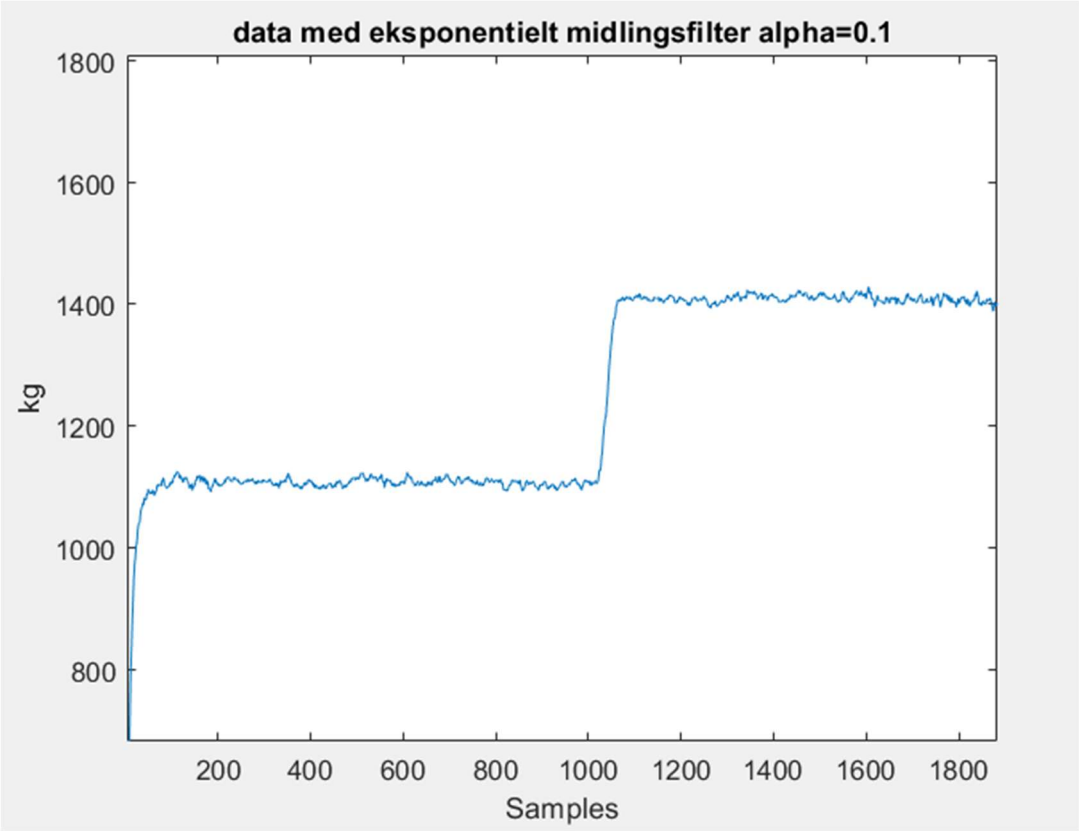
Vi har designet vores filter ud fra kode [3], som ses nedenfor.

Der er implementeret eksponentielle midlingsfiltre med tre forskellige α-værdier: 0,1, 0,5 og 0,9.

2.4.1 Eksponentielt midlingsfilter med α = 0,1

```
%% Implementér et eksponentielt midlingsfilter med α = 0.1
alpha = 0.1;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha .* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end

plot(y);  % plot filtreret med eksponentielt midlingsfilter (alpha = 0.1)
title('data med eksponentielt midlingsfilter alpha=0.1');
ylabel('kg');
xlabel('Samples');
```

Figur 14 koden for plot eksponentielt midlingsfilter med α-værdien på 0.1.

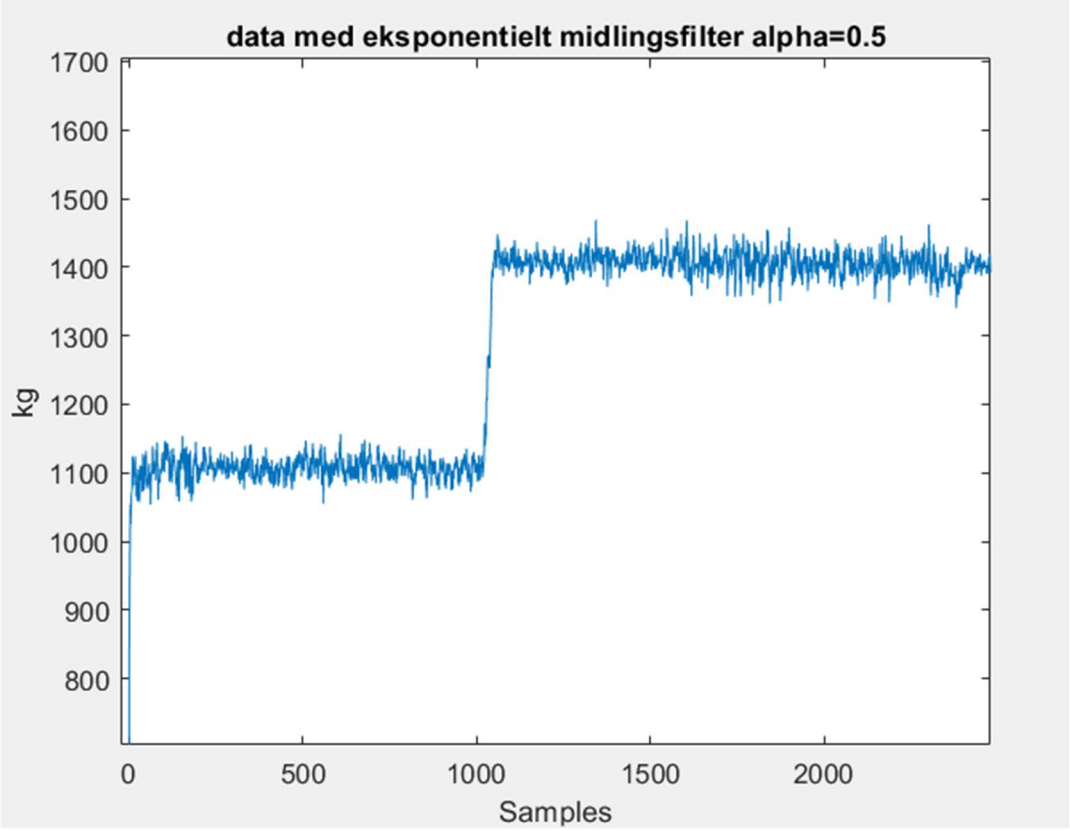


Figur 15 plot eksponentielt midlingsfilter med α-værdien på 0.1

2.4.2 Eksponentielt midlingsfilter med α = 0,5

```
%% Implementér et eksponentielt midlingsfilter med α = 0.5
alpha = 0.5;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha .* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end

plot(y);  % plot filtreret med eksponentielt midlingsfilter (alpha = 0.5)
title('data med eksponentielt midlingsfilter alpha=0.5');
ylabel('kg');
xlabel('Samples');
```

Figur 16 koden for plot eksponentielt midlingsfilter med α-værdien på 0.5.

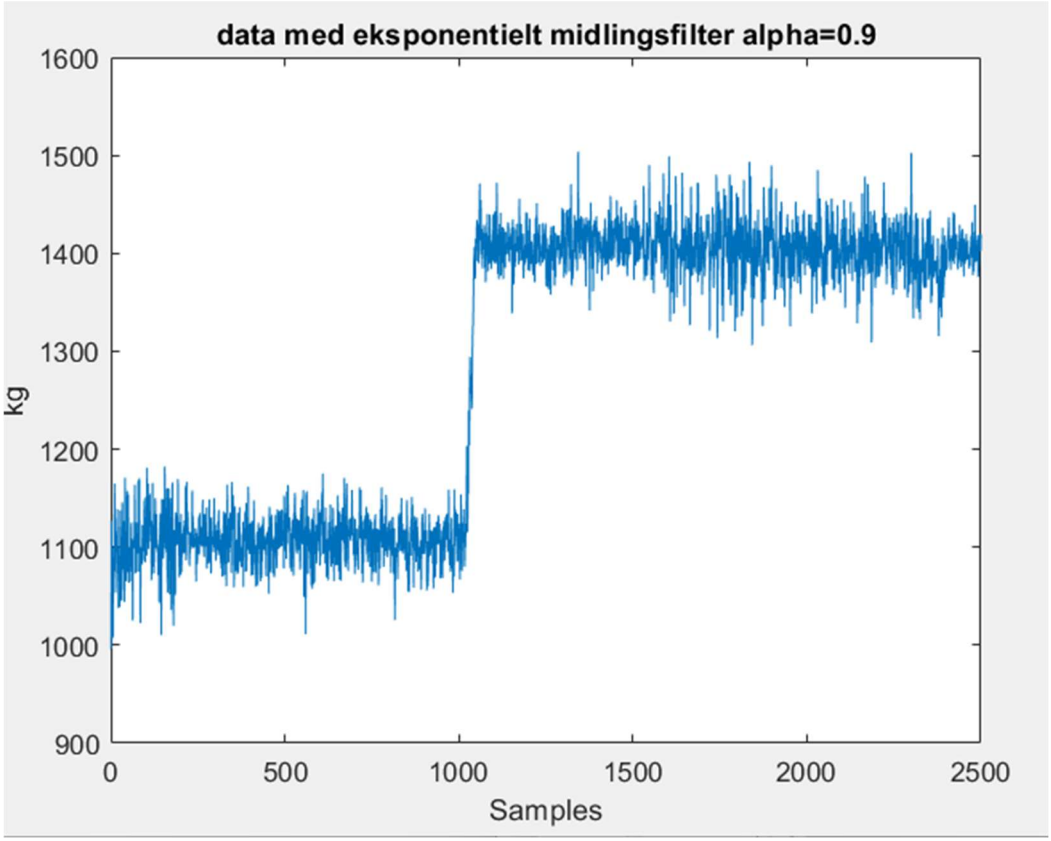


Figur 17 plot eksponentielt midlingsfilter med α-værdien på 0.5.

2.4.3 Eksponentielt midlingsfilter med α = 0,9

```
%% Implementér et eksponentielt midlingsfilter med α = 0.9
alpha = 0.9;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha .* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end

plot(y);  % plot filtreret med eksponentielt midlingsfilter (alpha = 0.9)
title('data med eksponentielt midlingsfilter alpha=0.9');
ylabel('kg');
xlabel('Samples');
```

Figur 18 koden for plot eksponentielt midlingsfilter med α-værdien på 0.9.



Figur 19 plot eksponentielt midlingsfilter med α-værdien på 0.9.

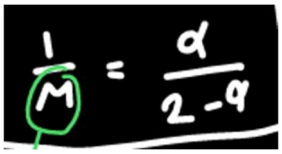
Hvis man ser på ovenstående figurer af det filtreerede signal ved forskellige værdier for koefficienten α kan man konkludere:

Lav α: langsom, men stabil respons med god støjfiltrering.

Høj α: hurtig respons, men større følsomhed for støj.

2.5: 100. ordens FIR midlingsfilter og α-værdien

Vi har beregnet α-værdien ud fra formlen [4] vist i Figur.



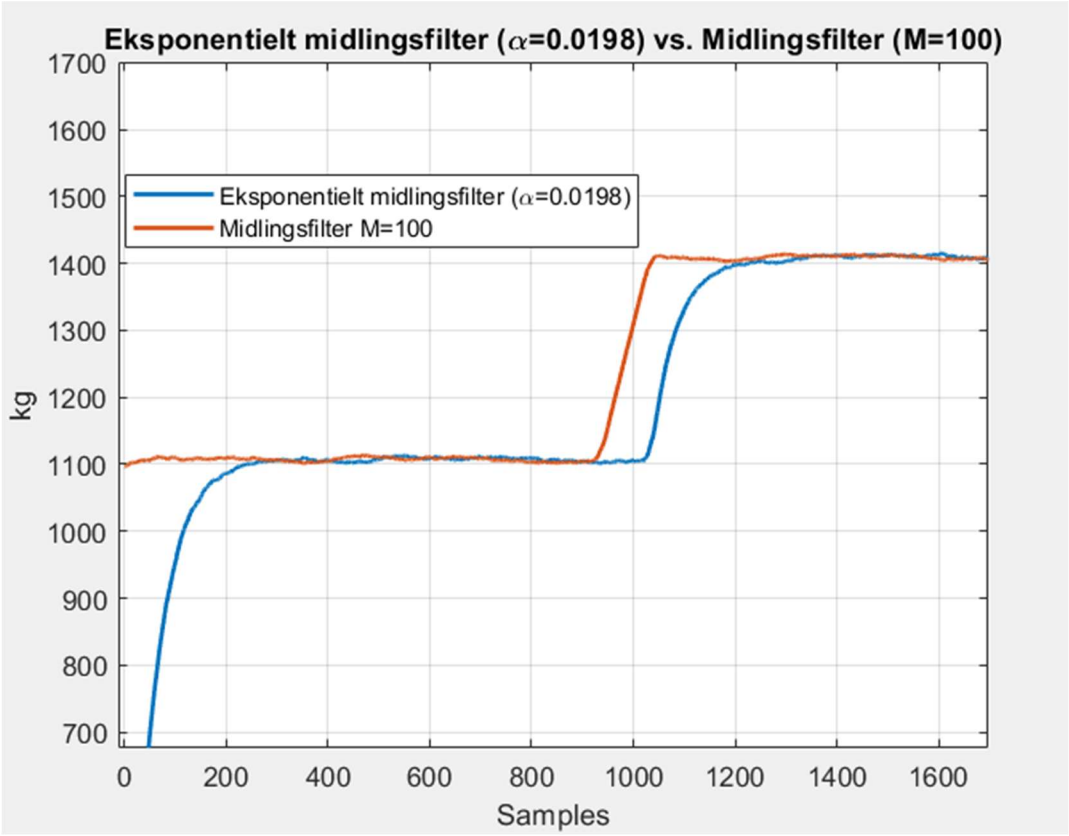
Figur 20 formlen for at finde α-værdien

$$M := 100$$

$$\alpha := \frac{1}{M} = \frac{\alpha}{2-\alpha} \xrightarrow{solve,\alpha} \frac{2}{101} = 0.019802$$

Resultatet viser, at for et FIR-midlingsfilter med en filterlængde på 100 fås en α-værdi på 0,0198.

For at sammenligne FIR-midlingsfilteret (N = 100) med det eksponentielle midlingsfilter (α = 0,0198), har vi plottet dem i samme graf.



Figur 21 plot eksponentielt midlingsfilter med a=0.0198 og Midlingsfiltre med 100 orden

De to filtre udfører næsten den samme filtrering, men det er tydeligt, at det eksponentielle midlingsfilter har en væsentligt længere afregningsperiode end FIR-midlingsfilteret med orden 100.

2.6: Korrupt data

Korrupt data refererer til samples, der afviger markant fra de øvrige, ofte pga. støj, målefejl eller andre anomalier. Fx kan enkelte samples i en tidsserie pludselig have en ekstremt høj værdi, som ikke passer til signalets mønster.

Konsekvenser:

- Forvrængning af signalet: Ekstreme værdier kan ændre det filtrerede signal og gøre fortolkning svær.
- Dårlig støjreduktion: Outliers øger variansen og forværrer støjen.

Løsning: Median-filter

Et median-filter er robust mod ekstreme værdier, i modsætning til gennemsnitsfiltre. Det sorterer samples i filtervinduet og vælger medianen (den højeste værdi for de nederste 50 % af samples'ne).

Eksempel:
For signalet: $x = [1\ 1\ 1\ 100\ 1\ 1\ 1]$

er værdien 100 et korrupt sample.

- Et gennemsnitsfilter ville forvrænge signalet ved at inddrage 100.
- Et median-filter vælger i stedet medianen (1), og signalet bevares uden forvrængning.

Opgave 3 - System overvejelser

Opgaven stiller et krav om, at det mindste ciffer i displayet for en vægt (der f.eks. går op til 5 kg) skal være en faktor 10 større end spredningen af støjen i signalet (dvs. standardafvigelsen, σ). Støjen er bestemt af filteret, som i dette tilfælde er et 100. ordens FIR-filter ($M = 101$).

For at bestemme det mindste ciffer, vil vi tage udgangspunkt i følgende udtryk og herefter beregne LSB:

$$LSB > \sigma_{efter\_filter} \cdot 10$$

Ligning 3 – Kravet for det mindstbetydende ciffer, LSB

$$\sigma_{efter\_filter} = \frac{\sigma_{før\_filter}}{\sqrt{M}}$$

Ligning 4– Formel for standardafvigelsen for signalet efter filtrering

$$\sigma_{efter\_filter} \cdot 10 = \frac{\sigma_{før\_filter} \cdot 10}{\sqrt{M}}$$

$$LSB > \frac{\sigma_{før\_filter} \cdot 10}{\sqrt{101}}$$

$$\frac{\sigma_{før\_filter} \cdot 10}{\sqrt{101}} \approx \sigma_{før\_filter}$$

$$LSB > \sigma_{før\_filter}$$

Hvis man derfor finder standardafvigelsen i det ufiltrerede signal, kan man derfor direkte bestemme hvad det mindstbetydende ciffer må være under de krav, der er stillet. Eksempelvis, hvis standardafvigelsen i signalet før filtreringen er mindre end et gram, så kan vi tillade os at sætte LSB til at være 1 g og stadig overholde kravet.

Konklusion

Vi har i denne case anvendt midlingsfiltre implementeret i Matlab på signaler fra en vejecelle. Vi har kunnet konstatere, i overensstemmelse med teorien, at variansen af støjen i signalet reduceres med en faktor på størrelse med længden af FIR midlingsfilteret (M). Man kan gennem denne form for filtrering opnå en mere præcis aflæsning af vægten.

Derudover implementerede vi også et eksponentielt midlingsfilter, som har den fordel at den kræver mindre memory-allokering, da den blot skal huske nuværende data input ($x[n]$) og det sidste output ($y[n-1]$), i modsætning til FIR-filteret, der har brug for at huske et antal af inputs, der er lige så stor som antallet af koefficienter (M).

Det er vigtigt altid at tage højde for støjen og dens varians i et pågældende system, da det er et udtryk for hvor pålidelige ens aflæsninger er. Arbejdet i denne case har vist, at man kan fjerne noget af denne støj ved at bruge et midlingsfilter på signalet.

Referencer.

[1] Midlingsfiltre 1 lektion slide fra brightspace.
https://brightspace.au.dk/content/enforced/183504-LR50157/GEK/DSA_Lek1_MidlingsfiltreI.pdf?isCourseFile=true&ou=183504

[2] Exponentielt Midlingsfiltre 2 lektion slide fra brightspace

https://brightspace.au.dk/content/enforced/183504-LR50157/DSA_lek2_MidlingsfiltreII1.pdf?ou=183504

[3] kode for eksponentielt midlingsfilter design.

https://brightspace.au.dk/d2l/le/lessons/183504/topics/2335689

[4] Midlingsfiltre 2 lektion slide fra brightspace

https://brightspace.au.dk/content/enforced/183504-LR50157/DSA_lek2_MidlingsfiltreII1.pdf?ou=183504

Appendix

```
%%CASE 1 - Indlæsning og opsætning af variable

clear; clc; close all;

S = load('vejecelle_data.mat');
x = S.vejecelle_data;

fs = 300; %Hz
N = length(x);
n = (0:N-1);
t = n/fs;

% tester


%% Opdeling af signal hhv belastet og ubelastet
plot(0:N-1,x);
%Ovenstående plot af rå data viser, at det transiente forløb stopper efter
%ca. 150 samples og overgangen fra belastet sker ved ca. 1000-1100 samples.
%Derfor tager vi forløbet med belastet vejecelle fra 150-1000 samples og
%ubelastet fra 1100-N
x_belastet = x(150:1000);
x_ubelastet = x(1100:N-1);

%% Opg 1 - Data analyse

mean_belastet = mean(x_belastet);
mean_ubelastet = mean(x_ubelastet);

var_belastet = var(x_belastet,1);
```

```
var_ubelastet = var(x_ubelastet,1);

std_belastet = std(x_belastet,1);
std_ubelastet = std(x_ubelastet,1);

fprintf('Belastet (1 kg):   mean = %.6g, var = %.6g, std = %.6g\n', mean_belastet, var_belastet,
std_belastet);
fprintf('Ubelastet (0 kg):  mean = %.6g, var = %.6g, std = %.6g\n\n', mean_ubelastet, var_ubelastet,
std_ubelastet);



%% Opg 1 – Histogrammer
% Histogrammer med ±1σ, ±2σ, ±3σ linjer
figure;

% ---------- Ubelastet ----------
subplot(1,2,1)
histogram(x_ubelastet, 'Normalization', 'pdf',NumBins=50); hold on; grid on
title('Ubelastet');
xlabel('Ukendt enhed'); ylabel('Sandsynlighedstæthed');

% Lodrette linjer ved ±1σ, ±2σ, ±3σ
mu = mean_ubelastet;
sigma = std_ubelastet;
xline(mu, 'k-', 'LineWidth', 1.2, 'DisplayName','mean');
xline(mu + sigma, 'r--', 'LineWidth', 1.2, 'DisplayName','±1σ');
xline(mu - sigma, 'r--', 'LineWidth', 1.2, HandleVisibility='off');
xline(mu + 2*sigma, 'g--', 'LineWidth', 1.2, 'DisplayName','±2σ');
xline(mu - 2*sigma, 'g--', 'LineWidth', 1.2, HandleVisibility='off');
xline(mu + 3*sigma, 'b--', 'LineWidth', 1.2, 'DisplayName','±3σ');
xline(mu - 3*sigma, 'b--', 'LineWidth', 1.2, HandleVisibility='off');
legend('show');

% ---------- Belastet ----------
subplot(1,2,2)
histogram(x_belastet, 'Normalization', 'pdf',NumBins=50); hold on; grid on
title('Belastet (1 kg)');
xlabel('Ukendt enhed'); ylabel('Sandsynlighedstæthed');

mu = mean_belastet;
sigma = std_belastet;
xline(mu, 'k-', 'LineWidth', 1.2, 'DisplayName','mean');
xline(mu + sigma, 'r--', 'LineWidth', 1.2, 'DisplayName','±1σ');
xline(mu - sigma, 'r--', 'LineWidth', 1.2, HandleVisibility='off');
xline(mu + 2*sigma, 'g--', 'LineWidth', 1.2, 'DisplayName','±2σ');
xline(mu - 2*sigma, 'g--', 'LineWidth', 1.2, HandleVisibility='off');
xline(mu + 3*sigma, 'b--', 'LineWidth', 1.2, 'DisplayName','±3σ');
xline(mu - 3*sigma, 'b--', 'LineWidth', 1.2, HandleVisibility='off');
```

```
legend('show');

%Histogrammerne for både den belastede og ubelastede tilstand viser en tydelig klokkeform, hvilket tyder på, at målesignalet (og støjen) er tilnærmelsesvist normalfordelt.

%Den målte spredning (standardafvigelse) stemmer også godt overens med histogrammets bredde:
%Flertallet af samples ligger indenfor ca. ±1 til ±2 standardafvigelser fra middelværdien, som forventet for en normalfordeling.

%% Opg 1 - PLot af frekvensspektret

% FFT af ubelastet signal – brug detrend til at fjerne offset/trend
Nf = length(x_ubelastet);            % antal samples

% Fjern både DC-offset og evt. langsom drift (lineær trend)
x_fft_u = detrend(x_ubelastet);
x_fft_b = detrend(x_belastet);

% Beregn FFT
X_u = fft(x_fft_u);
X_b = fft(x_fft_b);

% Frekvensakse (0 ... fs)
f = (0:Nf-1)*(fs/Nf);

% Plot kun den positive halvdel af spektret
figure;
subplot(2,1,1);
plot(f(1:floor(Nf/2)), abs(X_u(1:floor(Nf/2)))/Nf, 'LineWidth', 1.2);
xlabel('Frekvens [Hz]');
ylabel('Amplitude');
title('Frekvensspektrum af ubelastet signal (med detrend)');
grid on;

subplot(2,1,2);
plot(f(1:floor(Nf/2)), abs(X_b(1:floor(Nf/2)))/Nf, 'LineWidth', 1.2);
xlabel('Frekvens [Hz]');
ylabel('Amplitude');
title('Frekvensspektrum af belastet signal (med detrend)');
grid on;

%Spektret af det detrendede ubelastede signal fremstår bredbåndet og uden dominerende frekvenskomponenter, hvilket er karakteristisk for hvid støj.
%Selvom amplituden i spektret ikke er nul, er det netop forventeligt, da støjens energi er fordelt tilfældigt og jævnt over hele frekvensbåndet.
%Dette bekræfter, at den pålagte støj i signalet tilnærmelsesvist opfører sig som hvid støj.

%% Opg 1: Forskel i bit-niveau i gram
```

```
delta_counts = mean_ubelastet - mean_belastet;
LSB_g = 1000 / delta_counts;
LSB_kg = 1 / delta_counts;
fprintf('LSB = %.4f g/count (%.6f kg/count)\n\n', LSB_g, LSB_kg);



%% OPGAVE 2: MA-filter

% Variansanalyse af MA-filtrering
M_vals = [10, 50, 100];

% Variansanalyse af MA-filtrering
for M = M_vals
    y_b = moving_average_filter(x_belastet, M);
    y_u = moving_average_filter(x_ubelastet, M);

    y_b = y_b(M:end); y_u = y_u(M:end);
    var_b = var(y_b); var_u = var(y_u);

    fprintf('MA M=%d | Var belastet: %.4f | Var ubelastet: %.4f\n', M, var_b, var_u);
end

% Histogrammer for filtrerede signaler
figure;
for i = 1:3
    M = M_vals(i);
    y_u = moving_average_filter(x_ubelastet, M);
    subplot(2,3,i); histogram(y_u,400); title(sprintf('MA M=%d',M));
    axis([1300, 1500,0,600])
    y_b = moving_average_filter(x_belastet, M);
    subplot(2,3,i+3); histogram(y_b,400); title(sprintf('MA M=%d',M));
    axis([1000, 1200,0,300])
end

%Add titles for the rows
annotation('textbox', [0.048, 0.68, 0.8, 0.1], 'String', 'Ubelastet', ...
          'FontSize', 12, 'FontWeight', 'bold', 'EdgeColor', 'none', ...
          'HorizontalAlignment', 'left');

annotation('textbox', [0.05, 0.18, 0.8, 0.1], 'String', 'Belastet', ...
          'FontSize', 12, 'FontWeight', 'bold', 'EdgeColor', 'none', ...
          'HorizontalAlignment', 'left');



%% Moving-average (midlingsfilter) med filterlængde 100
M = 100;                    % filterlængde
```

```
N = numel(x);

xmem = zeros(M-1,1);
y    = zeros(N,1);

for nn = 1:N
    % Gennemsnit af (M-1) tidligere + den aktuelle prøve
    y(nn) = (sum(xmem) + x(nn)) / M;

    xmem = [x(nn); xmem(1:end-1)];
end

% Drop opstartsdel (de første M-1 samples er delvist udfyldt)
y_100M = y(M:end);

figure;
plot(y_100M, 'LineWidth', 2);
title('data med midlingsfilter (M = 100)');
ylabel('kg');
xlabel('Samples');
grid on;



%% Beregn den maksimale af FIR-midlingsfilteret:

% Definer den ønskede maksimale indsvingningstid i sekunder
max_indsvingningstid_sec = 0.1; % 100 millisekunder

% Beregn det tilsvarende antal samples baseret på samplingsfrekvensen
max_indsvingningstid_samples = max_indsvingningstid_sec * fs;

% Beregn den maksimale længde af FIR-midlingsfilteret
max_fir_length = floor(max_indsvingningstid_samples);

disp(['Den maksimale længde af FIR-midlingsfilteret: ', num2str(max_fir_length), ...
    'Samples']);




%% Implementér et eksponentielt midlingsfilter med α = 0.1
alpha = 0.1;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha .* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end
```

```
subplot(3,1,1);  % Create a subplot for the first filter
plot(y);  % plot filtreret med eksponentielt midlingsfilter (alpha = 0.1)
title('data med eksponentielt midlingsfilter alpha=0.1');
ylabel('kg');
xlabel('Samples');

% Implementér et eksponentielt midlingsfilter med α = 0.5
alpha = 0.5;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha .* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end

subplot(3,1,2);  % Create a subplot for the second filter
plot(y);  % plot filtreret med eksponentielt midlingsfilter (alpha = 0.5)
title('data med eksponentielt midlingsfilter alpha=0.5');
ylabel('kg');
xlabel('Samples');

% Implementér et eksponentielt midlingsfilter med α = 0.9
alpha = 0.9;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha .* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end

subplot(3,1,3);  % Create a subplot for the third filter
plot(y);  % plot filtreret med eksponentielt midlingsfilter (alpha = 0.9)
title('data med eksponentielt midlingsfilter alpha=0.9');
ylabel('kg');
xlabel('Samples');


%% α-værdien, således at får samme støj-reduktion, som for 100. ordens FIR midlingsfilter
alpha = 0.0198;
yold  = 0;

for n = 1:length(x)
    y(n) = alpha.* x(n) + (1 - alpha) .* yold;
    yold = y(n);
end

figure; clf
plot(y,    'LineWidth', 1.6); hold on; grid on
```

```
plot(y_100M,'LineWidth', 1.6);
title('Eksponentielt midlingsfilter (\alpha=0.0198) vs. Midlingsfilter (M=100)')
ylabel('kg'); xlabel('Samples');
legend('Eksponentielt midlingsfilter (\alpha=0.0198)','Midlingsfilter M=100','Location','best');
hold off
%%
```