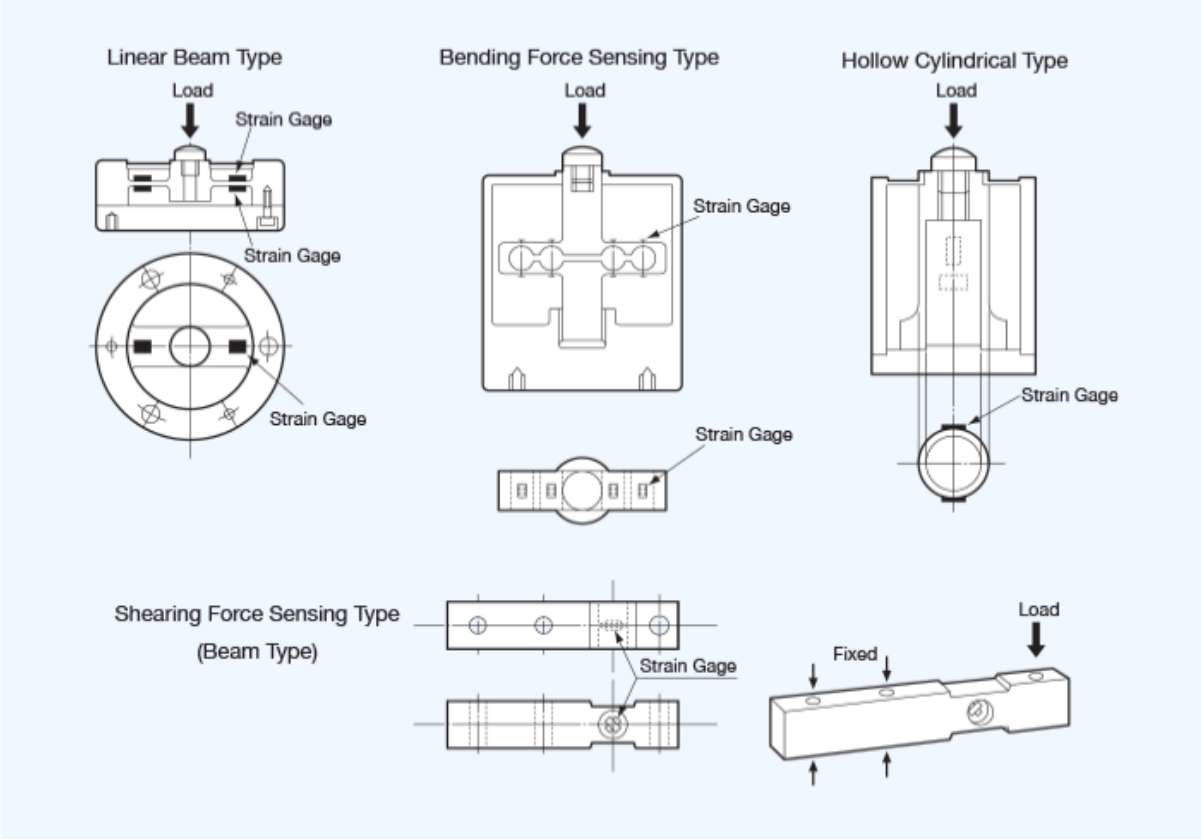
E4DSA Case 3 - Midling af sensordata

Janus Bo Andersen [1]

15. april 2020



[1] ja67494@post.au.dk

Indhold

1. Indledning

Tredje case i E4DSA er behandling og analyse af støjfyldt data fra en fysisk sensor. I dette tilfælde en vejecelle, dvs. en transducer fra kraftbelastninger til elektriske signaler. En vejecelle installeres, så påvirkning er enten fra kompression eller tension. På forsiden vises forskellige typer af vejeceller [1].

Fokus i denne case er at:

- Fjerne uønsket støj. Dette gøres vha. midlingsfiltre (lavpas).
- Kvantificere måleusikkerhed ved deskriptive statistiske mål.

Det er et hyppigt problem, at sensormålinger er overlejret med støj. Kilderne kan være mange:

- Processtøj: Ændringer i temperatur, lufttryk, vibrationer i omgivelserne, osv. under optagelse af målinger.
- Instrumentets måleusikkerhed: Drevet at sensitivitet i den analoge del af sensoren. Ved brug af en vejecelle over længere tid sker desuden et "drift" i output over tid [2, s. 1].
- Non-lineariteter og non-ideelle komponenter.
- Eletromagnetisk støj, feltkoblet eller ledningsbåret: Vil forstyrre optagelse af målinger.
- Kvantiseringsstøj i A/D converter: Effektiv outputstøj er kombinationen af forstærkning, samplingsfrekvens og ADC'ens opløsning [2, s. 2, fig. 4].

Reduktion af uønsket støj er nødvendigt for at få et bedre estimat på underliggende data og variable. Kvantificering af usikkerhed i målinger kan evt. henføres til forskellige kilder, såfremt der antages modeller for hhv. signalkomponenter og støj.

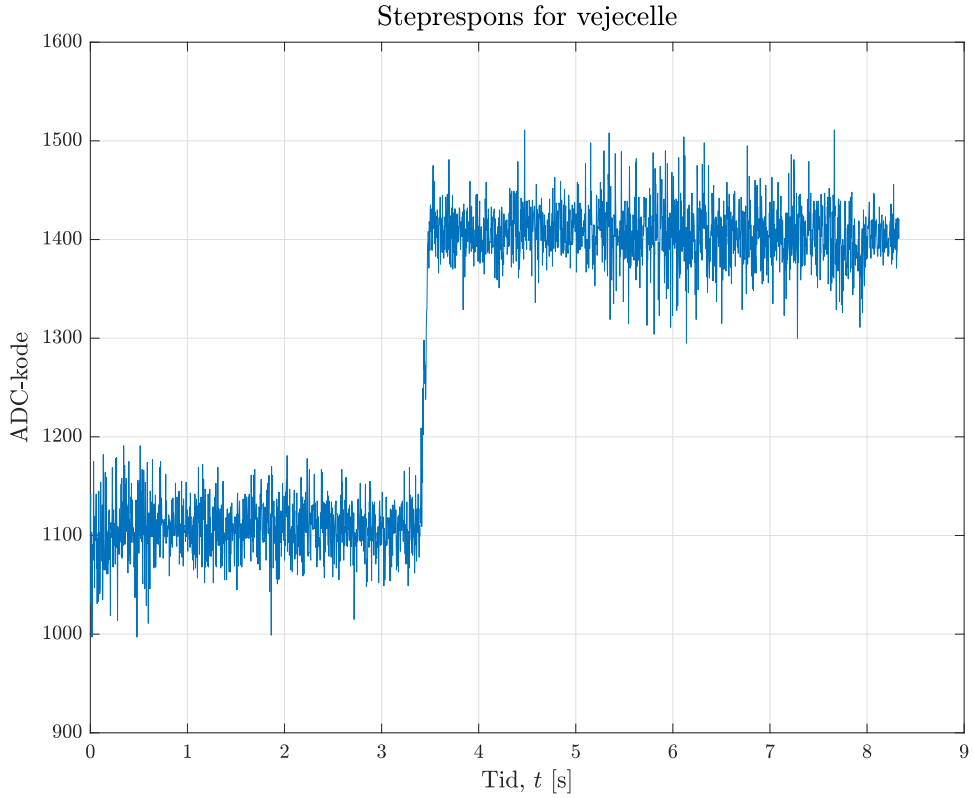
Der er yderligere motivation til støjreduktion, hvis data bruges videre i en regulator eller et feedback-kontrolsystem. Dér vil støj forringe kontrolkarakteristikken.

2. Opgave 1: Analyse af inputsignal

Inputsignalet er data optaget fra en vejecelle med samplingsfrekvensen 300 Hz. Data er ikke omregnet til en masseenhed, dvs. data er de rå ADC-koder. Vejecellen påvirkes med et stepinput: Belastes med en masse på 1 kg. Rå måledata vises i figuren nedenfor.

```
clc; clear all; close all;
setlatexstuff('latex');
load('vejecelle_data.mat');
x = vejecelle_data;
```

```
N = length(x);
t_vec = (0:N-1)/fs;
figure();
plot(t_vec, x); grid on;
xlabel('Tid, $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);
title('Steprespons for vejecelle', 'FontSize', 14);
```



Figuren viser, at kraftpåvirkningen sker ved ca. 3.5 s. Ubelastet tilstand er ca. ADC-koden 1100. Vi ved ikke noget om kalibrering af vejecelle inden måling, så det antages, at dette niveau er 0 kg. Belastet tilstand er ca. ADC-koden 1400, svarende til 1 kg. Der er væsentlig støj i data. Rise-time fra zero-state tager ca. 70 ms. Pga. støj kan evt. højere ordens indsvingning efter stepinput ikke ses.

2.1 Adskilning af tilstande

Her implementeres en kort algoritme til at detektere niveauskifte. Antag, at stepinput sker ved tiden $t = t^*$, svarende til $n = n^*$. Antag at støjen (målefejl) er normalfordelt som $\epsilon(n) \sim i.i.d.N(0, \sigma)$. Altså uafhængige, identisk fordelte (i.i.d.) stokastiske variable med varianshomogenitet[1]. Antag, at støjen er additiv. Antag også, at et støjfrit signal fra ADC'en ville være ren DC. Så er $x(n) = \mu_0 + \epsilon(n)$ for $n = 0 \ldots n^*$ og $x(n) = \mu_1 + \epsilon(n)$ for $n = n^* + 1 \ldots N - 1$. μ_0 og μ_1 er DC-amplituder i hhv. zero-state og med belastning. Så længe systemet er i zero-state skal 99 pct. af alle observationer ligge i intervallet $\hat{\mu}_0 \pm z_{\alpha/2} \cdot \hat{\sigma}_n$, hvor fx $z_{1\%/2} = 2.58$ er 99.5 pct.-fraktilen i standardnormal-fordelingen[2]. Når mere end et par observationer i træk er uden for intervallet, så må systemets tilstand være skiftet.

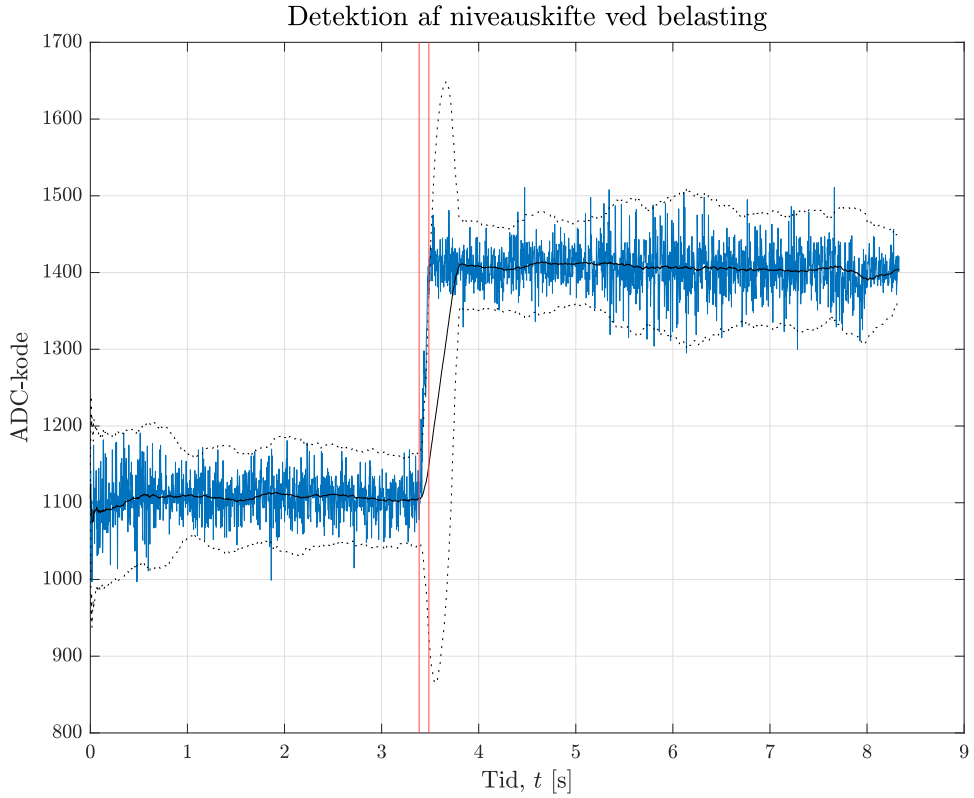
[1] D.s.s. homoskedasticitet, spredningen er uafhængig af n og μ_i.
[2] Her burde man have brugt den lidt bredere T-fordeling, fordi vi bruger estimat på både middelværdi $\hat{\mu}$ og varians $\hat{\sigma}^2$.

μ_0, μ_1 og σ estimeres vha. glidende gennemsnit. Den samlede model kan skrives:

$$x(n) = \mu_0(u(n) - u(n - n^*)) + \mu_1 u(n - n^*) + \epsilon(n)$$

Hvor $u(n)$ er stepinput, n^* er det ukendte steptidspunkt og $\epsilon \sim N(0, \sigma)$ er gaussisk støj.

```
MA_len_bk = 100;                    % benyt 100 historiske obs. til MA og SD
MA_len_fw = 0;                      % ingen fremtidige obs.
percentile = 1.0 - 1.0/100/2;       % 99.5 pct. fraktil (0.5 pct hver side)
sd = movstd(x, [MA_len_bk MA_len_fw]);  % beregn sigma(n)
ma = movmean(x, [MA_len_bk MA_len_fw]); % beregn estimat på A_0 el. A_1
ub = ma + norminv(percentile)*sd;       % Statistisk øvre grænse
lb = ma - norminv(percentile)*sd;       % Statistisk nedre grænse
t_trans = 0.1;                       % Anslået transienttid [s]


num_above = 3;                       % Når 3 obs. i træk ligger over => step
tstar = find((movmean((x > ub), [num_above 0]) >= 1), 1, 'first') / fs;

figure();
plot(t_vec, x); grid on; hold on;
plot(t_vec, ub, 'k:', t_vec, lb, 'k:');
plot(t_vec, ma, 'k-');
xline(tstar-t_trans/2, 'r'); xline(tstar+t_trans/2, 'r');
hold off;
xlabel('Tid, $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);
title('Detektion af niveauskifte ved belasting', 'FontSize', 14);
```



Figuren viser, at zero-state er t.v. for første vertikale røde markering. Belastet tilstand (steady state) er t.h. for anden vertikale røde markering. Mellem de røde markeringer er transientresponset. Denne opdeling benyttes i de følgende spørgsmål.

Figuren bekræfter også, at spredningen tilnærmet er homoskedastisk (ekskl. transientrespons).

```
N0end = (tstar-t_trans/2)*fs;        % 50 ms på hver side af t*
N1begin = (tstar+t_trans/2)*fs;
x0 = x(1:N0end);                     % dataserie for ubelastet vejecelle
x1 = x(N1begin:end);                 %  -- || --      belastet  -- || --
```

2.2 Q1.1. Deskriptiv statistik for de to tilstande

Deskriptiv statistik regnes nedenfor. Disse funktioner benytter stikprøve-beregning (Bessel-justering, division med $N-1$ i stedet for N) for at få en unbiased estimator. Estimater og estimatorer er[3]:

- Middelværdi, $\hat{\mu}$: $\bar{x} = \frac{\sum_{n=0}^{N-1} x(n)}{N-1}$.
- Varians, $\hat{\sigma}^2$: $s^2 = \frac{\sum_{n=0}^{N-1} (x(n)-\bar{x})^2}{N-1}$. Dvs. gennemsnitlig effekt i støjen.
- Standardafvigelse (spredning), $\hat{\sigma}$: $s = \sqrt{s^2}$. Dvs. RMS-værdi for støjen.

[3]Benyttet notation er således: En estimator, fx $\bar{x}$ eller s^2, er en matematisk funktion af en eller flere stokastiske variabler til at bestemme en ukendt parameter. Givet en fordeling af stokastiske inputvariable, findes en sandsynlighedsfordeling for estimatoren. En estimator kan være unbiased (middelret) eller biased. Et estimat, fx $\hat{\mu}$ eller $\hat{\sigma}^2$, er en enkelt realisering på en tilnærmelse til den sande parameterværdi. Givet et sæt observationer er estimatet deterministisk. Indhentes et nyt sæt observationer fås et nyt, muligvis anderledes, estimat. Osv.

Beregninger er foretaget med MATLABs funktioner.

```
means = [round(mean(x0)) round(mean(x1))];      % mu_hat
stddevs = [std(x0) std(x1)];                    % sigma_hat
vars = stddevs.^2;                              % sigma_hat^2
T = table(means', stddevs', vars', [length(x0) length(x1)]');
T.Properties.RowNames = {'x_0', 'x_1'};
T.Properties.VariableNames = {'Mid', 'Std', 'Var', 'N'};
disp(T)
```

```
          Mid      Std       Var       N
          ____    ______    ______    ____
    x_0   1106    27.593    761.36    1016
    x_1   1406    28.072    788.05    1455
```

Tabellen viser middelværdier for ADC-koderne i de to tilstande, svarende til belastning med hhv. 0 kg og 1 kg. Det giver ingen mening at have decimaler på disse værdier (ADC-kode er heltal). Tabellen viser også, at standardafvigelserne er tilnærmelsesvis ens. Det er altså rimeligt fortsat at antage, at støjen i de to dele af signalet er fordelt med samme spredningsparameter σ. Man kunne evt. udføre en hypotesetest (F-test) for at teste det statistisk, men det er uden for scope her.

2.3 Q1.2. Histogrammer

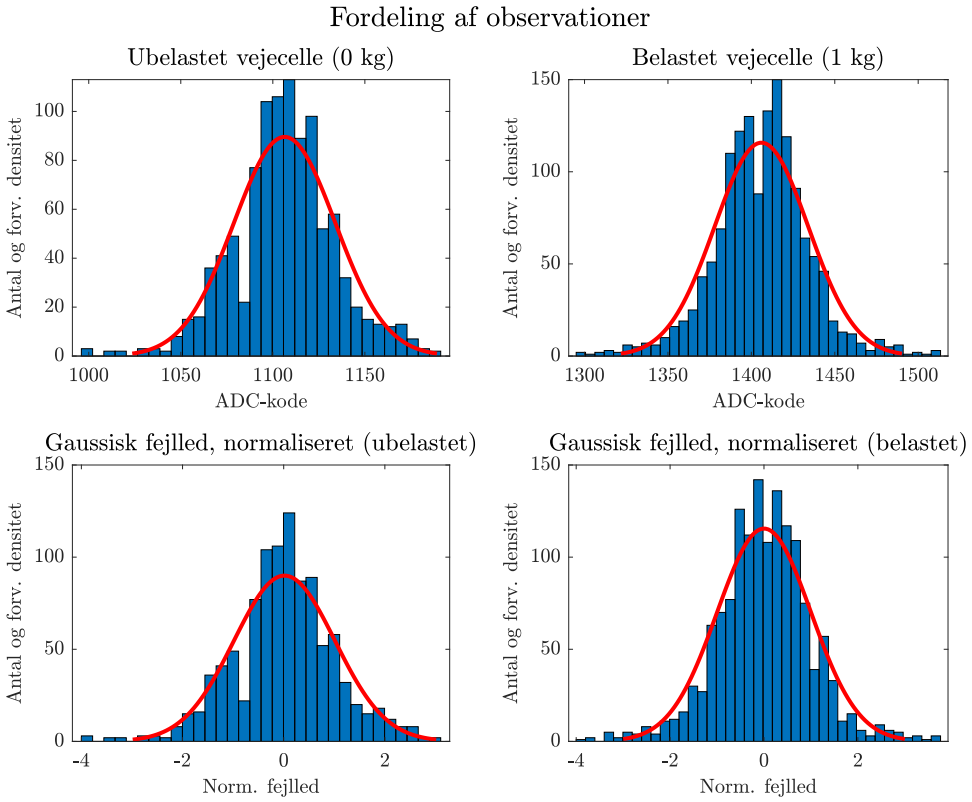
Det er rimeligt at arbejde videre med antagelsen, at modellerne er $x_0(n) = \mu_0 + \epsilon(n)$ og $x_1(n) = \mu_1 + \epsilon(n)$, hvor $\epsilon \sim N(0, \sigma)$. Dvs. vi har to stationære stokastiske processer $x_i \sim N(\mu_i, \sigma)$, hvor $i = \{0, 1\}$ angiver enten zero-state eller belastet tilstand.

Det betyder, at hele måleusikkerheden betragtes udelukkende som stationær gaussisk støj nu. Det er naturligvis ikke helt sandt[4]. Nedenfor undersøges antagelsen om normalfordeling nærmere ved at plotte histogrammer samt fittede tæthedsfunktioner, under antagelsen at data er normalfordelt. Observerede fejlled, $\hat{\epsilon}(n) = x(n) - \hat{\mu}$, normaliseres til $z(n) = \frac{\hat{\epsilon}(n)}{\hat{\sigma}}$ og plottes imod en standardnormalfordeling. Det er igen en tilsnigelse, for ADC-koderne er heltalsværdier.

```
figure();
sgtitle('Fordeling af observationer', 'FontSize', 14);
subplot(221)
histfit(x0);
title('Ubelastet vejecelle (0 kg)', 'FontSize', 12);
xlabel('ADC-kode', 'FontSize', 10);
ylabel('Antal og forv. densitet', 'FontSize', 10);
subplot(222)
histfit(x1);
title('Belastet vejecelle (1 kg)', 'FontSize', 12);
xlabel('ADC-kode', 'FontSize', 10);
ylabel('Antal og forv. densitet', 'FontSize', 10);
```

[4] Det er jo summen af kvantiseringsfejl, instrumentets målefejl, processtøj, osv.

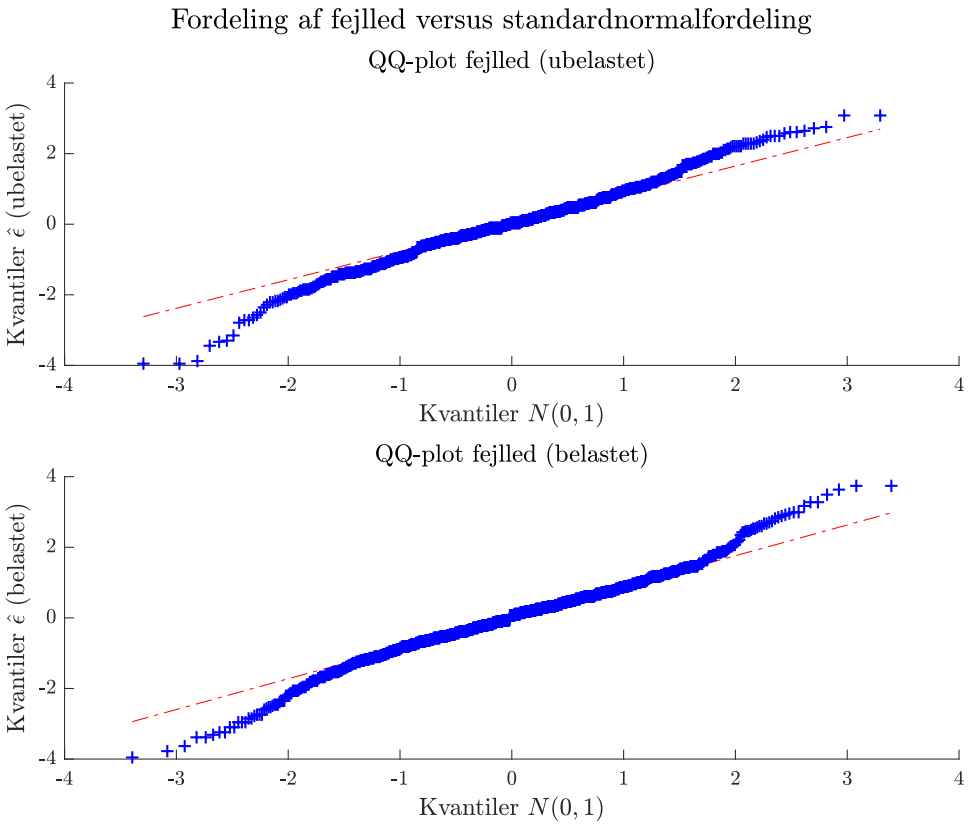
```
subplot(223)
histfit( (x0-means(1))/stddevs(1) );
title('Gaussisk fejlled, normaliseret (ubelastet)', 'FontSize', 12);
xlabel('Norm. fejlled', 'FontSize', 10);
ylabel('Antal og forv. densitet', 'FontSize', 10);
subplot(224)
histfit( (x1-means(2))/stddevs(2) );
title('Gaussisk fejlled, normaliseret (belastet)', 'FontSize', 12);
xlabel('Norm. fejlled', 'FontSize', 10);
ylabel('Antal og forv. densitet', 'FontSize', 10);
```



Det ser tilnærmelsesvist normalfordelt ud. For at få en lidt mere robust konklusion, suppleres ovenstående figurer med QQ-plots, dvs. fordelingssammenligning af normaliserede observationer ($z(n) = \frac{x(n)-\hat{\mu}}{\hat{\sigma}}$) versus forventede fraktiler givet en standardnormalfordeling. Ved en perfekt normalfordeling af data, ville alle punkter i plottet ligge på en ret linje, oven i den røde linje.

```
figure();
sgtitle('Fordeling af fejlled versus standardnormalfordeling', ...
    'Interpreter', 'Latex', 'FontSize', 14);
subplot(211)
qqplot((x0-means(1))/stddevs(1));
title('QQ-plot fejlled (ubelastet)', 'FontSize', 12);
xlabel('Kvantiler $N(0,1)$', 'FontSize', 12);
ylabel('Kvantiler $\hat{\epsilon}$ (ubelastet)', 'FontSize', 12);
subplot(212)
```

```
qqplot((x1-means(2))/stddevs(2));
title('QQ-plot fejlled (belastet)', 'FontSize', 12);
xlabel('Kvantiler $N(0,1)$', 'FontSize', 12);
ylabel('Kvantiler $\hat{\epsilon}$ (belastet)', 'FontSize', 12);
```



QQ-plots bekræfter, hvad der kunne anes i histogrammer: Fordelingen af målefejl har "fede haler", dvs. en bredere fordeling med flere observationer langt fra middelværdien, end man ville forvente givet normalfordelingen. Dog viser både histogrammer og QQ-plots, at fordelingen stadig kan betragtes som tilnærmelsesvist normal. Så det er rimeligt at konkludere, at støjled (fejlled) er tilnærmelsesvist normalfordelte, og at de samlede signaler er ligeså. Vi kan derfor godt, uden at begå store regnefejl, benytte antagelsen, at måleusikkerheder på målinger foretaget med vejecellen vil være normalfordelte i både belastet og ubelastet tilstand.

Hvis vi havde behov for at være sikre, ville det give mening at foretage et hypotesetest, fx vha. et Kruskal-Wallis-test.

2.4 Q1.3. Effektspektra (spektraltæthed) for hvid støj

Hvid støj forståes som et signal, der har samme intensitet af alle frekvenser, eller mere præcist: Hvid støj har konstant spektraltæthed (PSD) for alle frekvenser, praktisk set dog kun inden for en vis båndbredde. De fejlled $\epsilon \sim i.i.d.N(0,\sigma)$ vi har analyseret tidligere er en type hvid støj [3]: Additiv gaussisk hvid støj.

Spektral estimation benyttes til at estimere et effektspektrum for en stationær stokastisk proces, der per definition har en periode på $T \to \infty$. Det giver et estimat på gennemsnitlig effekt over et frekvensbånd.

Principielt divergerer Fourier-integralet for en uendelig lang hvid støj-proces ($T \to \infty$)[5]. Se estimation gøres med gentagen sampling af udsnit med finit længde T fra den stokastiske proces. Der bruges så en estimator, som *vil* eksistere i grænsen $T \to \infty$:

- For en enkelt samplingperiode regnes: $\frac{|X_T(f)|^2}{T}$
- Som middelværdi (forventet værdi) efter gentagen sampling: $\mathrm{E}\{\frac{|X_T(f)|^2}{T}\}$
- I grænsen $T \to \infty$: $S_{xx}(f) = \mathrm{PSD}_{xx}(f) = \lim_{T\to\infty} \mathrm{E}\{\frac{|X_T(f)|^2}{T}\}$
- I diskret tid med DFT [4]: $S_{xx}(f) = \frac{(\Delta t)^2}{T}|X(f)|^2$. Med $\Delta t = \frac{1}{f_s}$ og $T = N\Delta t$, bliver ensidet skalerinsfaktor $\frac{2}{N \cdot f_s}$.

Det er et avanceret emne. Her estimeres med kun en enkelt sampleperiode, inspireret af [5]. Alternativt kunne man have delt den tilgængelige data op i flere segmenter og lavet gentagen estimation, som ved STFT.

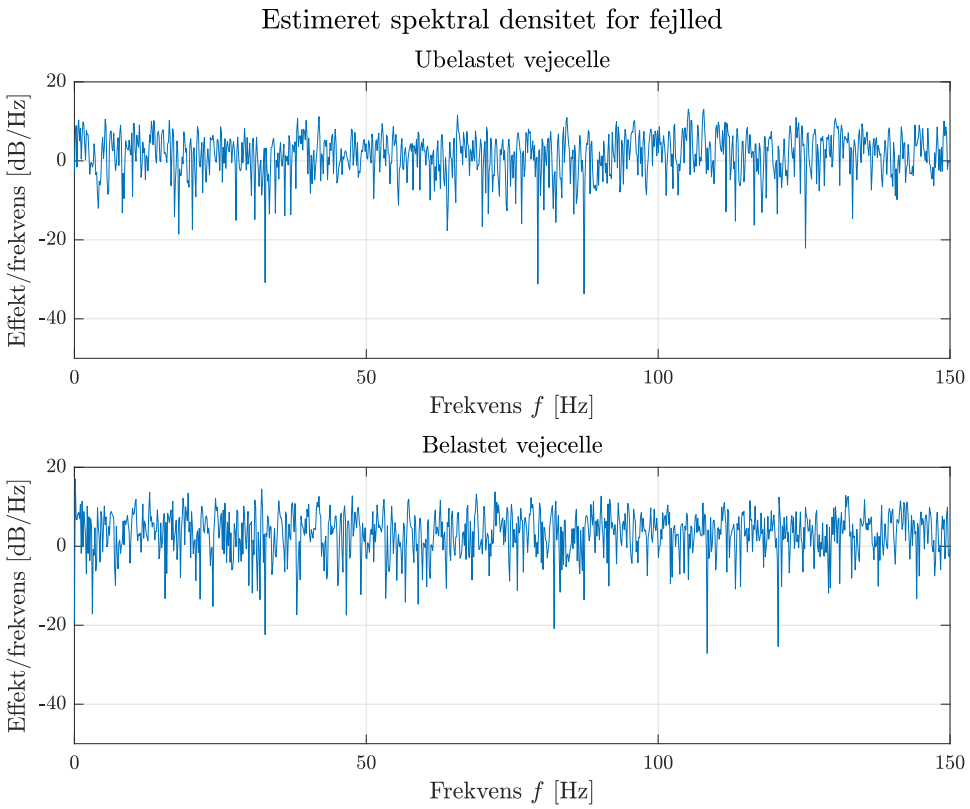
DC-komponenten i sensordata er her uinteressant og trækkes ud. Dvs. vi kigger kun på spektralindhold i fejlled: $\hat{\epsilon}_i(n) = x_i(n) - \hat{\mu}_i$, hvor $i = \{0, 1\}$ som tidligere angiver enten zero-state eller belastet tilstand.

Med samplingsfrekvens på 300 Hz tillader samplingssætningen at kigge på spektral densitet op til Nyquist-frekvensen 150 Hz.

```
e0 = x0 - means(1);     % fejlled for målinger i ubelastet tilstand
e1 = x1 - means(2);     %      -- || --           belastet tilstand

Nfft = 2^nextpow2( max( length(e0), length(e1) ));     % Zero-padding
psd_scale = 2/(fs*Nfft);              % Skalering for at approximere PSD
f_vec = (0:Nfft-1)*(fs/Nfft);                          % Frekvensvektor

E0 = fft(e0, Nfft);     % Beregn FFT'er (zero-padded)
E1 = fft(e1, Nfft);

E0S = psd_scale * (E0 .* conj(E0));    % Effektspektrum og skalering
E1S = psd_scale * (E1 .* conj(E1));

figure();
sgtitle('Estimeret spektral densitet for fejlled', 'FontSize', 14);
subplot(211)
plot(f_vec, 10*log10(E0S) );
xlim([0 fs/2]); ylim([-50 20]); grid on;
title('Ubelastet vejecelle', 'FontSize', 12);
xlabel('Frekvens $f$ [Hz]', 'FontSize', 12);
ylabel('Effekt/frekvens [dB/Hz]', 'FontSize', 12);
subplot(212)
plot(f_vec, 10*log10(E1S) );
xlim([0 fs/2]); ylim([-50 20]); grid on;
```

[5] Additiv gaussisk hvid støj har uendelig energi over intervallet $-\infty$ til $+\infty$, så er ikke i L^2, dvs. $\int_{t=-\infty}^{\infty} |f(t)|^2 dt$ divergerer og eksisterer ikke. I diskret tid vil det sige, at $\sum_{n=-\infty}^{\infty} |f(n)|^2$ er uendelig.

```
title('Belastet vejecelle', 'FontSize', 12);
xlabel('Frekvens $f$ [Hz]', 'FontSize', 12);
ylabel('Effekt/frekvens [dB/Hz]', 'FontSize', 12);
```



Figuren viser, at spektraltætheden er tilnærmelsesvis konstant for alle frekvenser. Det er ensbetydende med, at støjen er hvid støj. Denne konklusion gælder for både ubelastet og belastet tilstand. Det er frekvensdomænets sidestykke til at vi tidligere erklærede målefejlene for $i.i.d.N(0, \sigma)$.

2.5 Q1.4. ADC-opløsning

Frekvensopløsning i en ADC (eller DAC) afgøres af værdiområde ift. antal niveauer i enkodering af I/O spænding. For en N-bit ADC er opløsningen $\frac{V_{max}-V_{min}}{2^N}$ [6, 12-7, s. 634].

Vi kender ikke værdiområdet for ADC'en og ej heller antallet af bits. Men, vi ved hvor mange niveauer i ADC-kode, der kræves til at dække intervallet fra 0 kg til 1 kg. Dvs. vi kan beregne opløsningen i gram ved:

$$\text{resolution} = \frac{\Delta\text{masse i [g]}}{\Delta\text{ADC-kode}}$$

```
d_mass = 1000 - 0;                              % gram [g]
d_ADC_niv = round(means(2)) - round(means(1));  % 1406 - 1106 = 300
resolution = round(d_mass / d_ADC_niv,2);       % 3.33 g/niv.

disp([ num2str(d_ADC_niv) ' ADC-niv. ml. 0 [g] og 1000 [g]' newline ...
      '=> afstand ml. bit-niv. ' num2str(resolution) ' [g/niv.]' ]);
```

```
300 ADC-niv. ml. 0 [g] og 1000 [g]
=> afstand ml. bit-niv. 3.33 [g/niv.]
```

Givet dette resultat, er værdien af LSB i ADC'en 3.33 g. Hvis vi antager, at det er en 16-bit lineær ADC, kan det fulde værdiområde estimeres ved en lineær ekstrapolation med lidt algebra, hvor λ er ADC-kode og $f(\lambda)$ er vægt i gram: $f(\lambda) - f(\lambda_0) = b_1(\lambda - \lambda_0)$, som også skrives $f(\lambda) = b_0 + b_1\lambda$, med $b_0 = f(\lambda_0) - b_1\lambda_0$. Hældningen b_1 er opløsningen, vi regnede ovenfor.

```
lam_0 = round(means(1));          % 1106 (ADC-kode, ubelastet)
f_lam_0 = 0;                      % 0.0 [g]
b_1 = resolution;                 % hældning (opløsning)
b_0 = f_lam_0 - b_1*lam_0;        % skæring
lam_max = 2^16 - 1;               % maks. ADC-kode (65535)
lam_min = 0;                      % min. ADC-kode (0)

f_max = b_0 + b_1*lam_max;
f_min = b_0 + b_1*lam_min;

disp(['Fuld ADC-bitbredde svarer til ' num2str(round(f_min)) ...
     ' [g] til ' num2str(round(f_max)) ' [g].' ]);
```

```
Fuld ADC-bitbredde svarer til -3683 [g] til 214549 [g].
```

Så under antagelsen, at strain-gauge i vejecellen er lineær i hele intervallet mellem -3.6 kg og 214.5 kg, så svarer vejecellens funktionsområde til en udvidet personvægt :) Selvom en strain-gauge sikkert også virker ved negativ kraftpåvirkning, så giver det umiddelbart kun mening at have det negative interval, så der er noget "spillerum" til digitalt enten at kalibrere vejecellen til 0.0 kg eller sætte "tare".

3. Opgave 2: Design af midlingsfilter

For at reducere støjen i signalet fra vejecellen, dvs. opnå et bedre estimat på den sande måleværdi, eksperimenteres her med midlingsfiltre (MA-filtre). Som set ovenfor, kan et glidende gennemsnit benyttes som en rullende estimator på en middelværdi jf. de store tals lov, da $\bar{x}$ vil konvergere mod μ.

Ved udtagning af N stikprøver af en normalfordelt stokastisk variabel med *ukendt varians*, som er tilfældet for x, bruges den centrale grænseværdisætning til at opstille et $1-\alpha$-konfidensinterval for den sande middelværdi μ. Med sikkerhed på $1-\alpha$ kan siges, at parameteren er indeholdt i følgende interval [7, s. 90]:

$$\mu = \bar{x} \pm t_{N-1,\alpha/2} \cdot \frac{s}{\sqrt{N}}$$

Hvor s er estimator på standardafvigelsen for x, $\frac{s}{\sqrt{N}}$ er standardfejlen, dvs. std.afv. på $\hat{\mu}$ og $t_{N-1,\alpha/2}$ er en to-sidet fraktil fra T-fordelingen med $N-1$ frihedsgrader.

Det væsentlige her er, at for $N \to \infty$ konvergerer usikkerheden på estimatet mod 0. Dvs. variansen på estimatet konvergerer mod 0. Jo længere vi laver MA-filteret, jo sikrere et estimat får vi. Formuleret anderledes; Givet inputvarians på data ind i filteret på $\text{var}(x) = \hat{\sigma}^2_{\text{input}}$, så bliver variansen på output fra filteret:

$$\text{var}(\hat{\mu}) = \hat{\sigma}^2_{\text{output}} = \frac{\hat{\sigma}^2_{\text{input}}}{N}$$

Et MA(100)-filter ville fx teoretisk give 100 gange dæmpning af støjens AC-effekt.

Omkostningen er længere indsvingningstid grundet flere delays. Transientrespons er først "slut", når filterets delay line er fyldt op med *aktuelle* værdier. Så for et MA(N)-filter er transientresponset $N-1$ samples langt, svarende til $t_{\text{trans}} = \frac{N-1}{f_s}$.

3.1 Q2.1. Forskellige midlingsfiltre

Nedenfor opstilles FIR (ikke-rekursive) midlingsfiltre med længderne $N = 10$, $N = 50$ og $N = 100$.

Filtrene designes i tidsdomænet via impulsresponset, dvs. koefficienterne i foldningssummen for MA(N):

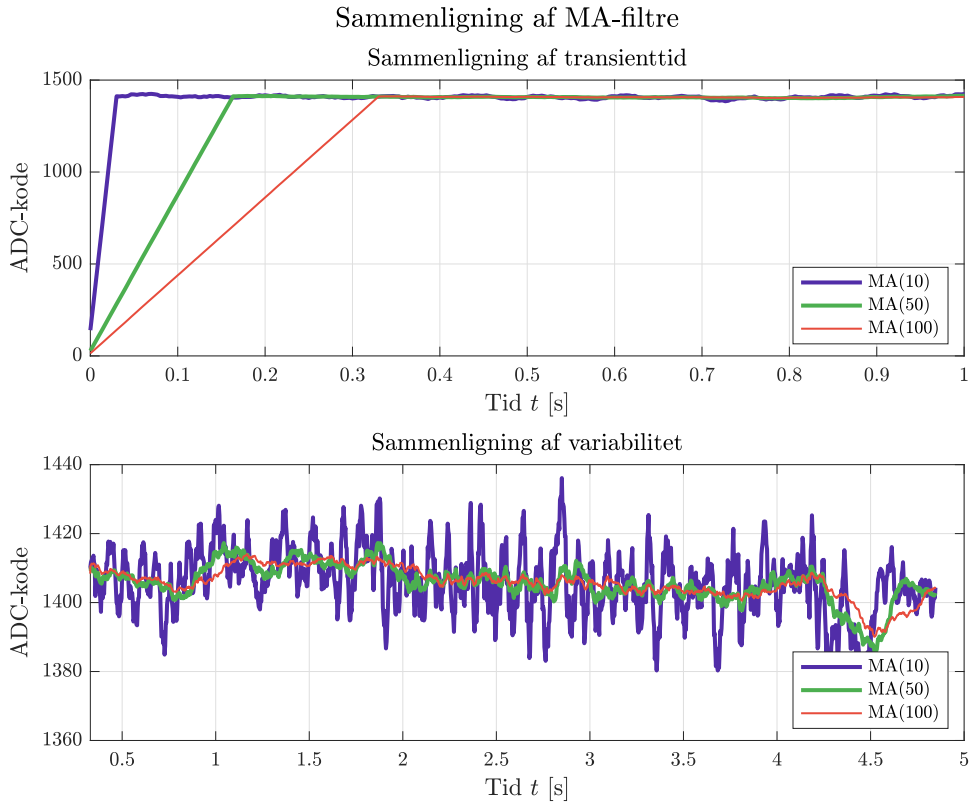
$$y_{\text{MA}(N)}(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(n-k)$$

Typisk vil $x(n)$-værdier med negativt indeks udgå (erstattes med nul), hvilket giver en meget markant indsvingning. MATLABs `movmean`-funktion justerer i stedet filterlængden fra 0 til N, som tilgængelige samples stiger. Det giver en mindre tydelig/stejl indsvingning.

Filtrering foretages på data fra belastet vejecelle, $x_1(n)$ for at få estimaterne $\hat{\mu}_1$ og $\hat{\sigma}^2$.

```
h_MA10 = ones([1 10])/10;                % MA(10)
h_MA50 = ones([1 50])/50;                % MA(50)
h_MA100 = ones([1 100])/100;             % MA(100)

y_MA10 = filter(h_MA10, 1, x1);          % Filtrering af x1
y_MA50 = filter(h_MA50, 1, x1);
y_MA100 = filter(h_MA100, 1, x1);

N = length(x1);
t_vec = (0:N-1)/fs;

f = figure();
sgtitle('Sammenligning af MA-filtre', 'FontSize', 14);
s1 = subplot(211);
p1 = plot(t_vec, y_MA10, 'Color', [81, 45, 168]/255, 'LineWidth', 2);
grid on; hold on;
p2 = plot(t_vec, y_MA50, 'Color', [76, 175, 80]/255, 'LineWidth', 2);
p3 = plot(t_vec, y_MA100, 'Color', [231, 76, 60]/255, 'LineWidth', 1);
hold off;
```

```
xlim([0 1]);
legend({'MA(10)', 'MA(50)', 'MA(100)'}, 'Location', 'southeast');
title('Sammenligning af transienttid', 'FontSize', 12);
xlabel('Tid $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);

s2 = subplot(212);
copyobj([p3 p2 p1], s2);
grid on; box on; xlim([0.33 5]);
legend({'MA(10)', 'MA(50)', 'MA(100)'}, 'Location', 'southeast');
title('Sammenligning af variabilitet', 'FontSize', 12);
xlabel('Tid $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);
```



Første figur viser tydeligt den væsentlige forskel i indsvingningstid for forskellige filterlængder. Anden figur viser reduktion i variabilitet for længere MA-filtre, som forventet.

Det ses på figuren, at der efter ca. 4 s er en længerevarende sekvens af outliers. Disse punkter er tydelige outliers ift. standardnormalfordelingen, så de skyldes sandsynligvis ikke normal målefejl / støj. Det kunne skyldes berøring af vejecellen. Denne data fjernes i de videre sammenligninger.

For at lave en fair sammenligning af filtrenes evne til at reducere støj, regnes standardafvigelser efter fuldendt indsvingning for det langsommeste filter, MA(100), dvs. fra $n = 100$, $t = 0.33$.

```
nb = 100;   % n_begin, fuldendt indsving., n=Filterlængde => t=n/fs=0.33
```

```
ne = 1200;  % n_end, t=4.0 => n=t*fs=1200
bm = var(x1(nb:ne));   % ufiltreret benchmark for effekt i støj

% Regn varians (effekt) af støj i filtrerede signaler
var_errs = [var(y_MA10(nb:ne)), var(y_MA50(nb:ne)), var(y_MA100(nb:ne))];

% Regn reduktion af effekt (variansreduktions-faktor)
err_red = bm ./ var_errs;

% Reduktion i dB
err_red_dB = 10*log10(err_red);

% Forventede reduktioner
err_red_exp = [10 50 100];
err_red_exp_dB = 10*log10(err_red_exp);

rows = {'MA(10)', 'MA(50)', 'MA(100)'}';
columns = {'Varians', 'Eff_reduk', 'Forv_reduk', 'Reduk_dB', 'Forv_dB'};
T = table(var_errs', err_red', err_red_exp', ...
          err_red_dB', err_red_exp_dB');
T.Properties.RowNames = rows;
T.Properties.VariableNames = columns;
disp(T)
```

	Varians	Eff_reduk	Forv_reduk	Reduk_dB	Forv_dB
MA(10)	82.728	9.951	10	9.9787	10
MA(50)	16.372	50.282	50	17.014	16.99
MA(100)	11.36	72.466	100	18.601	20

Tabellen viser, at filtrene tilnærmet giver den teoretisk forventede reduktion i støjeffekt for MA(10) og MA(50). Det er ikke helt tilfældet for MA(100). Det skyldes nok bl.a. outlieren ved omkring 2.8 s, der jo relativt set påvirker MA(100)-filteret i længere tid end de to andre filtre.

3.2 Q2.2. Maksimal indsvingningstid

Som nævnt ovenfor er indsvingningstid for et N-tap FIR-filter givet ved $t_{\text{trans}} = \frac{N-1}{f_s}$. Et muligt krav til maksimaltid er 100 ms. Så for $t_{\text{trans, max}} = 0.1$ løses for N:

```
t_transmax = 0.1;               % 100 ms
N_max = t_transmax * fs + 1;    % Maks. antal tappe

disp(['Maksimalt antal tappe for at overholde maks. transientrespons: ' ...
       num2str(N_max) '.']);
```

```
Maksimalt antal tappe for at overholde maks. transientrespons: 31.
```

Dette resultat stemmer fint overens med figuren, da $N = 31$ ligger ca. i midtpunktet mellem MA(10) og MA(50), og midtpunktet af deres transienttid er omkring 0.1 s.

Man kunne også implementere filteret rekursivt, så det får differensligningen

$$y(n) = \frac{1}{N}(x(n) - x(n-N)) + y(n-1)$$

Indsvingningstiden bliver naturligvis den samme, fordi der stadig kræves N delays for at holde $x(n-N)$.

3.3 Q2.3. Eksponentielt midlingsfilter

Man kan bruge et eksponentielt midlingsfilter (IIR) i stedet. Det er rekursivt. Fordelene er, at karakteristikken nemt kan justeres i real-tid, og at filtrering kræver færre beregninger og færre memory-elementer. Differensligningen er

$$y(n) = \alpha x(n) + (1-\alpha)y(n-1)$$

Hvor $0 < \alpha < 1$. Overføringsfunktionen er [6, 11-33, s. 612]

$$H(z) = \frac{\alpha}{1-(1-\alpha)z^{-1}}$$

Det kan vises, at dæmpning af støjeffekten følger [6, 11-29, s. 610]

$$\frac{\hat{\sigma}^2_{\text{output}}}{\hat{\sigma}^2_{\text{input}}} = \frac{\alpha}{2-\alpha}$$

Så en given værdi af α giver følgende faktor støjreduktion, R:

$$R = \frac{2-\alpha}{\alpha}$$

Tilsvarende, givet en ønsket dæmpning af støj med faktor R, kan α findes [6, 11-30, s. 611]:

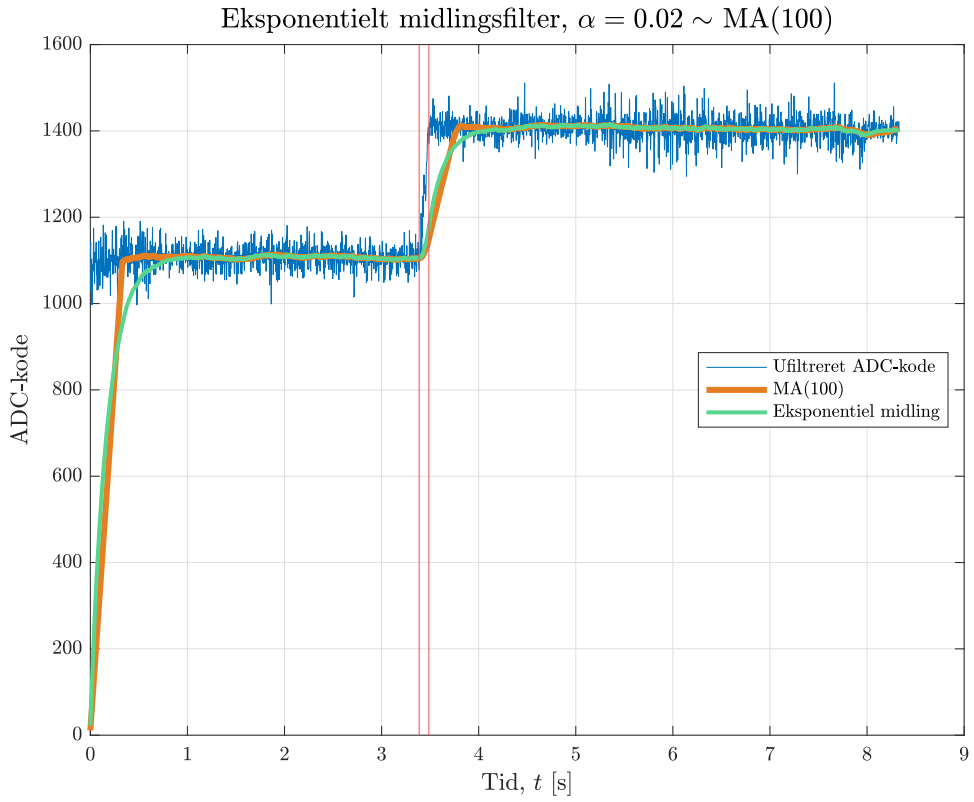
$$\alpha = \frac{2}{R+1}$$

Hvis man vil have hurtig indsvingning kan man jo starte med α tæt på 1. Gradvist kan støjdæmpning så øges ved at reducere α.

Et MA(100)-FIR-filter, der teoretisk dæmper støj med faktor $R = 100$, kan emuleres som følger:

```
1  R = 100;
2  alpha = 2/(R+1);          % Alpha: støjreduktion som et MA(100) FIR
3
4  % Overføringsfunktion
5  b = alpha;
6  a = [1 -(1-alpha)];
7
8  y_exp = filter(b,a,x);      % Filtrering af hele signalet, inkl. steps
9  y_MA100 = filter(h_MA100, 1, x);
10
```

```
N = length(x);                  % Hele signallængden
t_vec = (0:N-1)/fs;

figure();
plot(t_vec, x); grid on; hold on;
plot(t_vec, y_MA100, 'Color', [230, 126, 34]/255, 'LineWidth', 3);
plot(t_vec, y_exp, 'Color', [88, 214, 141]/255, 'LineWidth', 2);
xline(tstar-t_trans/2, 'r'); xline(tstar+t_trans/2, 'r');
hold off;
xlabel('Tid, $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);
tit = 'Eksponentielt midlingsfilter, $\alpha=0.02 \sim$ MA(100)';
title(tit, 'Interpreter', 'latex', 'FontSize', 14);
legend({'Ufiltreret ADC-kode', 'MA(100)', 'Eksponentiel midling'}, ...
    'Location', 'east');
```



Figuren viser, at det nye filter i hastighed er langsommere end MA(100). Den rigtige sammenligning er nok på stigetid, fx. fra 10 pct. til 90 pct. af stephøjden. Det langsomme repsons er fordi α er sat så lavt.

Den anden vigtige del af testen er selvfølgelig på støjreduktion. Samme metode benyttes som ved MA-filtrene. Grundet at eksponential-filteret er langsommere, forskydes starttidspunktet for testen. Hvis dette ikke gøres, har eksponentialfilteret en meget ringere gennemsnitlig dæmpning af effekt fra støj (vi regner jo varians på en del transientresponset så).

```
nb = 250;
```

```
bm = var(x1(nb:ne));              % Genberegn benchmark

% Genberegn filtrering på kun x1
y_exp100 = filter(b,a,x1);
y_MA100 = filter(h_MA100, 1, x1);

% Genberegn med R=10
R = 10; alpha = 2/(R+1);
b10 = alpha; a10 = [1 -(1-alpha)];

y_exp10 = filter(b10,a10,x1);
y_MA10 = filter(h_MA10, 1, x1);

% Sammenlign!
var_errs = [var(y_exp100(nb:ne)), var(y_MA100(nb:ne)) , ...
            var(y_exp10(nb:ne)), var(y_MA10(nb:ne))];

err_red = bm ./ var_errs;
err_red_dB = 10*log10(err_red);
err_red_exp = [100 100 10 10];
err_red_exp_dB = 10*log10(err_red_exp);

columns = {'Varians', 'Eff_reduk', 'Forv_reduk', 'Reduk_dB', 'Forv_dB'};
T = table(var_errs', err_red', err_red_exp', ...
          err_red_dB', err_red_exp_dB');
T.Properties.RowNames = {'exp(a=0.02)', 'MA(100)', 'exp(a=0.18)', 'MA(10)'
    };
T.Properties.VariableNames = columns;
disp(T)
```

	Varians	Eff_reduk	Forv_reduk	Reduk_dB	Forv_dB
exp(a=0.02)	13.197	66.802	100	18.248	20
MA(100)	12.48	70.64	100	18.491	20
exp(a=0.18)	85.921	10.26	10	10.112	10
MA(10)	86.964	10.137	10	10.059	10

Tabellen viser, at filtrene under "steady state" fungerer nogenlunde som forventet. Støjdæmpningen for det første eksponentielle midlingsfilter er marginalt dårligere end sammenligneligt MA-filter, grundet det uendelige impulsrespons. Samme udfordring som før ses, hvor de længere filtre påvirkes relativt mere af enkelte outliers. Det er værst for IIR-filteret.

Der kan allerede drages tre konklusioner om denne filtertype:

- Prisen for færre beregninger og memory-elementer er et langsommere filter og marginalt dårligere støjdæmpning.
- Et eksponentielt filter (IIR) er mere følsomt over for outliers end et FIR-filter, fordi det principielt

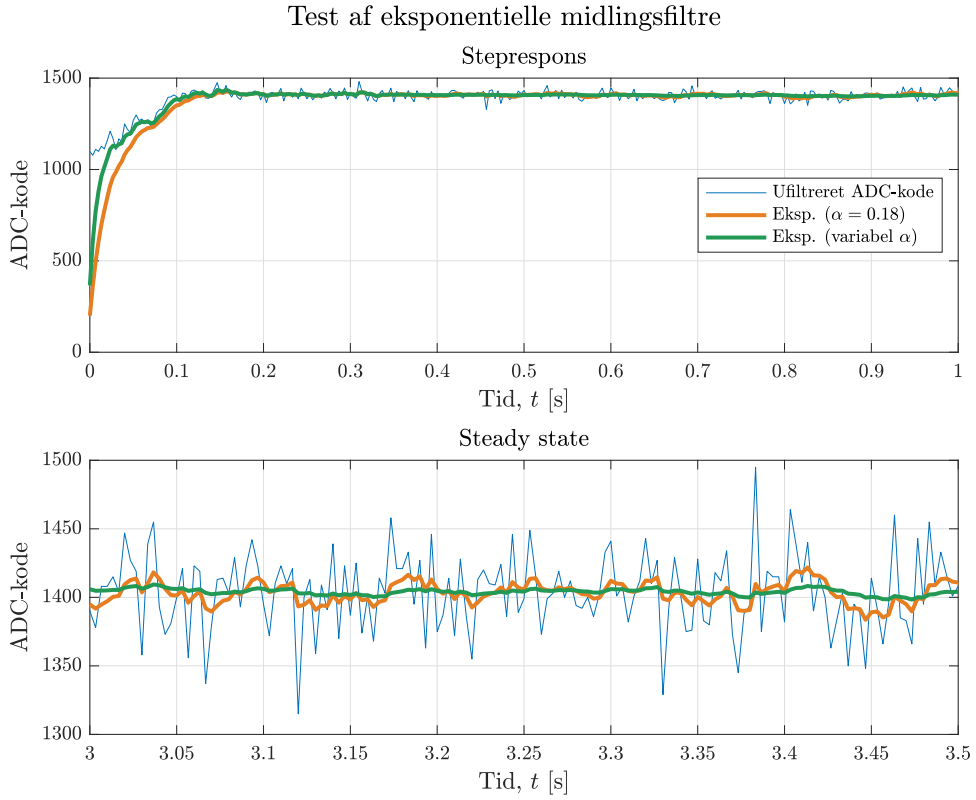
påvirkes uendeligt af impulser. Jo kraftigere outliers, jo værre. Jo lavere α, jo længere tid er effekten fra en outlier om at "dø ud".

- Når man har betalt "prisen", så bør man også udnytte den fleksibilitet, man får fra filteret, navnlig at α kan justeres.

Herunder sammenlignes respons fra 2 eksponentielle midlingsfiltre: Filter med $\alpha = 0.18$, svarende til MA(10), som netop er testet, og et nyt, der består af 3 sektioner, hvor α løbende justeres.

```
% Datasæt med step
xs = x(N0end:end);
N = length(xs);
t_vec = (0:N-1)/fs;

% Del 1-3
R = 5; alpha1 = 2/(R+1);            % alpha = 0.33
R = 10; alpha2 = 2/(R+1);           % alpha = 0.18
R = 50; alpha3 = 2/(R+1);           % alpha = 0.04

% startværdibetingelser
y_exp = zeros([1 N]);               % pre-allocate
dy = 0;                             % delay line

% De første n=50 filtreres med del 1
alpha = alpha1;                     % alpha for denne sektion
nstart = 1; nend = nstart + 50;
for n = nstart:nend
    y = alpha*xs(n) + (1-alpha)*dy; % differensligning
    dy = y;                         % gem i delay line
    y_exp(n) = y;                   % gem nyeste output
end

% De næste n=50 filtreres med del 2
alpha = alpha2;                     % alpha for denne sektion
nstart = nend + 1; nend = nstart + 50;
for n = nstart:nend
    y = alpha*xs(n) + (1-alpha)*dy; % differensligning
    dy = y;                         % gem i delay line
    y_exp(n) = y;                   % gem nyeste
end

% De resterende filtreres med del 3
alpha = alpha3;                     % alpha for denne sektion
nstart = nend + 1; nend = N;
for n = nstart:nend
    y = alpha*xs(n) + (1-alpha)*dy; % differensligning
    dy = y;                         % gem i delay line
    y_exp(n) = y;                   % gem nyeste
```

```
end

% sammenligninger, filter med alpha=0.18, svarende til MA(10)
y_exp10 = filter(b10,a10,xs);

% Plot sammenligning
figure();

s1 = subplot(211);
p1 = plot(t_vec, xs); hold on; grid on;
p2 = plot(t_vec, y_exp10, 'Color', [230, 126, 34]/255, 'LineWidth', 2);
p3 = plot(t_vec, y_exp, 'Color', [34, 153, 84]/255, 'LineWidth', 2);
hold off;
xlim([0 1]);
title('Steprespons', 'FontSize', 12);
xlabel('Tid, $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);
legend({'Ufiltreret ADC-kode', ...
        'Eksp. ($\alpha=0.18$)', 'Eksp. (variabel $\alpha$)'}, ...
        'Location', 'east');

s2 = subplot(212);
copyobj([p3 p2 p1], s2);
grid on; box on; xlim([3 3.5]);
title('Steady state', 'FontSize', 12);
xlabel('Tid, $t$ [s]', 'FontSize', 12);
ylabel('ADC-kode', 'FontSize', 12);

sgti = 'Test af eksponentielle midlingsfiltre';
sgtitle(sgti, 'Interpreter', 'latex', 'FontSize', 14);
```



Ovenstående figurer viser styrken ved denne filtertype. Den grønne kurve viser, at man kan få "best of both worlds": Hvis man benytter en strategi, hvor α gradvist sænkes, så opnås et filter, der *både* stabiliseres hurtigt (kort steprespons) og har høj støjdæmpning (lav varians).

3.4 Q2.4. Manglende observationer, outliers

"Korrupt" data, fx outliers eller "missing values", betyder at filteret skal stabiliseres (indsvinges) igen, og at output fra filteret er "korrupt" i et stykke tid.

Nettobetydningen afhænger af filtertype og filterlængde, og er et trade-off i design af filteret:

- Langt filter (lav α-værdi): Betydning af en outlier er relativt lille ift. summen af alle de andre samples. Men, en "Black Swan"-outlier eller mange "missing values" vil betyde dårligt output i lang tid / mange samples. I et FIR-filter falder fejlen(e) ud på et tidspunkt. I et IIR-filter "lever" de videre for evigt.
- Kort filter (høj α-værdi): Filteret tilpasser sig hurtig igen, dvs. fejlen "dør ud" hurtigt. Til gengæld er betydningen en relativt meget større fejl i outputtet, fordi en enkelt dårlig værdi vægter meget i et kort filter.

Valg af filterlængde ift. korrupt data afhænger af systemets mulighed for at pre-processere data, fx fjerne outliers (real-tid eller ej) samt systemets krav til robusthed versus hurtig reaktion/tilpasning.

Fix: Det er vigtigt, at justeringer til signalet bibeholder en konstant samplingsfrekvens, så statistik og spektrum stadig kan regnes uden for meget bias. Mulige måder at fikse "korrupt" data er:

- Start/afslutning af signal kan trimmes væk.
- Et lille antal "Missing values" kan "erstattes": Fx med median, middelværdi eller interpolation.

- Med en model for data (fx en regressionsmodel), kan estimerede værdier indsættes.
- Hvis en større mængde data mangler, kan decimering/resampling benyttes, og lavere frekvenser vil da stadig være "tilgængelige" spektralt.

Der findes uden tvivl mere intelligente metoder inden for specifikke anvendelsesområder.

4. Opgave 3: Systemovervejelser

Det er interessant at designe systemet, så måleinstrumentet på et display kan udlæse et bestemt antal betydende, pålidelige cifre. Det er en helt oplagt designparameter til et målesystem.

Vi antager, at den "rå" målefejl (uanset kilden) er additiv og udviser varianshomogenitet (altså er uafhængig af målingens størrelse). Det forhold er allerede illustreret i tidligere afsnit. Vi ved så, at hvis måleværdien er en stationær stokastisk proces med tidsinvariant middelværdi (dvs. ingen step mens vi måler), så kan vi nedbringe variansen (fejlen) på måleestimatet ved at inkludere flere samples i beregning af middelværdien.

Designudfordringen er så at nedbringe spredningen på estimatet så tilpas meget, at instrumentet har den ønskede præcision. Vi tager nu udgangspunkt i et system, hvor designet allerede er fastlagt. Der er et MA(100)-filter, så vi ved, at

$$\hat{\sigma}_{\mathrm{MA}(100)} \approx \frac{\hat{\sigma}_{\mathrm{input}}}{\sqrt{100}}$$

Fra opgave 1.1 ved vi, at $\hat{\sigma}_{\mathrm{input}} = 28.1$ [ADC-koder]. Vi ved også fra opgave 1.4, at niveauer i ADC'en er 3.33 [g/ADC-kode]. Så enheden for spredningen på middelværdien kan regnes om til vores ønskede enhed til præsentation på displayet [kg]:

$$\hat{\sigma}_{\mathrm{MA}(100)} = \frac{\hat{\sigma}_{\mathrm{input}}}{\sqrt{100}}[\text{ADC-koder}] \cdot 3.33[\frac{\mathrm{g}}{\text{ADC-kode}}] \cdot 10^{-3}[\frac{\mathrm{kg}}{\mathrm{g}}] = 9.36 \cdot 10^{-3}[\mathrm{kg}]$$

Det vil altså sige en spredning på 9.36 g efter MA(100)-filteret.

Vi vil nu gerne sikre, at den udlæste måleværdi ligger inden for 10 standardafvigelser på estimatet, hvilket er en ekstremt høj grad af konfidens. For det mindst betydende ciffer på displayet skal gælde:

$$\mathrm{LSB} > 10 \cdot \hat{\sigma}_{\mathrm{MA}(100)} = 93.6 \cdot 10^{-3}[\mathrm{kg}]$$

Så hvis vi lader displayet vise måleresultater i steps af $100 \cdot 10^{-3}[\mathrm{kg}] = 0.1[\mathrm{kg}]$, dvs. 100 g, så holder vi os på den sikre side.

Desuden bør man implementere en algoritme til at undgå flicker på displayet, som forelået i [2, s. 5] :)

5. Konklusion

I denne case er der behandlet data fra en fysisk sensor, med fokus på reduktion af støj vha. midlingsfiltre og på at forstå og kvantificere støjen gennem simpel deskriptiv statistik.

Der er desuden lavet sammenligninger af forskellige typer midlingsfiltre og forskellige filterordener. Der er diskuteret fordele og ulemper ved hver type, og der er fremvist en løsning med et eksponentielt midlingsfilter med variabel parameter, hvor man kan "få det bedste fra begge verdener": Hurtig stigetid og høj dæmpning af støj.

Det har været en interessant og relevant case, med mange videre anvendelser i indlejret systemudvikling, reguleringsteknik, m.v.

6. Kildehenvisning

[1] Kyowa. *Load Cells (Load Transducers)*. 2020. URL: https://www.kyowa-ei.com/eng/technical/sensors/loadcells.html.

[2] Colm Slattery og Mariah Nie. "A Reference Design for High-Performance, Low-Cost Weigh Scales". I: *Analog Dialogue* 39.12 (dec. 2005).

[3] Wikipedia.org. *White Noise*. 2020. URL: https://en.wikipedia.org/wiki/White_noise.

[4] Wikipedia.org. *Spectral Density. Power Spectral Density*. 2020. URL: https://en.wikipedia.org/wiki/Spectral_density#Power_spectral_density.

[5] The Mathworks. *Power Spectral Density Estimates Using FFT*. 2020. URL: https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html.

[6] Richard G. Lyons. *Understanding Digital Signal Processing*. 3. udg. Prentice Hall, 2010.

[7] Steen Andersen, Peder Ostergaard og Steen Lund-Thomsen. *Notesamling til HA og HD Statistik*. H278. Institut for Marketing, Informatik og Statistik, Aarhus School of Business, 2004.

7. Funktioner

Der er til projektet implementeret en række hjælpefunktioner.

7.1 setlatexstuff

```
function [] = setlatexstuff(intpr)
% Sæt indstillinger til LaTeX layout på figurer: 'Latex' eller 'none'
% Janus Bo Andersen, 2019
    set(groot, 'defaultAxesTickLabelInterpreter',intpr);
    set(groot, 'defaultLegendInterpreter',intpr);
    set(groot, 'defaultTextInterpreter',intpr);
    set(groot, 'defaultGraphplotInterpreter',intpr);

end
```