

AARHUS UNIVERSITET

INTRODUKTION TIL DIGITAL SIGNALBEHANDLING

3. SEMESTER

DSA Case 2

Gruppemedlemmer:

Gustav A. Gammelgaard

Stinus Lykke Skovgaard

Tim Hede Stenholt

AU id:

au538293

au520659

au543518



1. april 2017

Indhold

1	Problem beskrivelse	3
2	Opgave 1 - Data analyse	3
3	Opgave 2 - Design af midlingsfilter	4
3.1	Lineært midlingsfilter	5
3.2	Eksponentielt midlingsfilter	8
4	Opgave 3 - System overvejelser	10
5	Matlab kode	10

Figurer

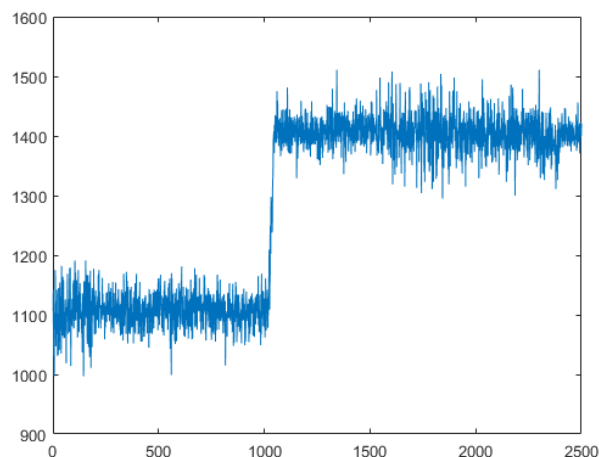
1	Sample akse for hele signalet	3
2	Effekt-spektrum for noload	4
3	Effekt-spektrum for load	4
4	Histogram af load med og uden filter. Filterorden = 10	5
5	Filter respons for det lineære midlingsfilter	6
6	Histogram af noload med og uden filter. Filterorden = 10	6
7	Histogram af load med og uden filter. Filterorden = 50	7
8	Histogram af load med og uden eksponentielt filter. $N = 10$	8
9	Histogram af noload med og uden eksponentielt filter. $N = 10$	9
10	Linært midlingsfilter respons på step ved $M=10$	9
11	Eksponentielt midlingsfilter respons på step ved $N=10$	10

1 Problem beskrivelse

Denne case omhandler måden at forbedre et støjfyldt signal med et midlingsfilter. Dette filter skal midle et signal fra en vægt. Typisk vil signalet fra vægten svinge voldsomt som vil gøre det svært at måle korrekt. Midlingsfilteret vil gøre det muligt at få en pålidelig måling. Filteret er realiseret ved hjælp af matlab og vægt data er importeret ind i matlab og testet i dette miljø.

2 Opgave 1 - Data analyse

Ved data analysen er det udleverede materiale blevet plottet som en sample tidsakse. Dette viser at vores signal har et område med og uden vægt på.



Figur 1: Sample akse for hele signalet

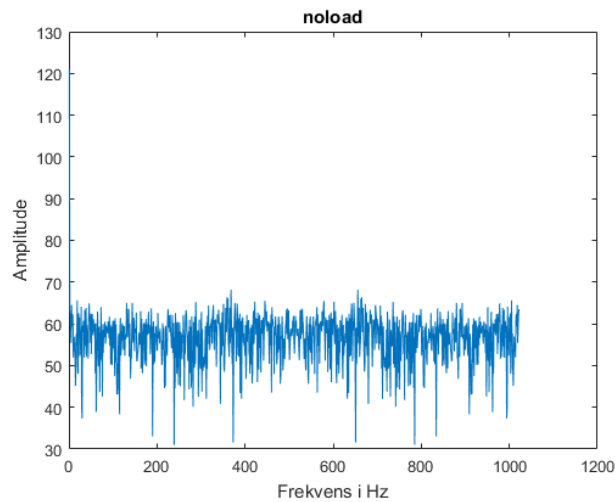
på figur 1 kan man se den belastede og ubelastede del af signalet. Da dataen skal bruges i et midlingsfilter undlader vi at bruge de samples der ligger i ændringen, da signalet først skal falde til ro. Derfor får vi to signaler der bliver kørt filtre på. Henholdsvis et load signal og noload signal.

Hvis dataen fra load og unload signalet plottes på et histogram kan man se hvordan dataen er normalfordelt. Hvis man udregner varians og spredning, får vi spredningen til ca 30. Det passer fint overens med histogrammerne.

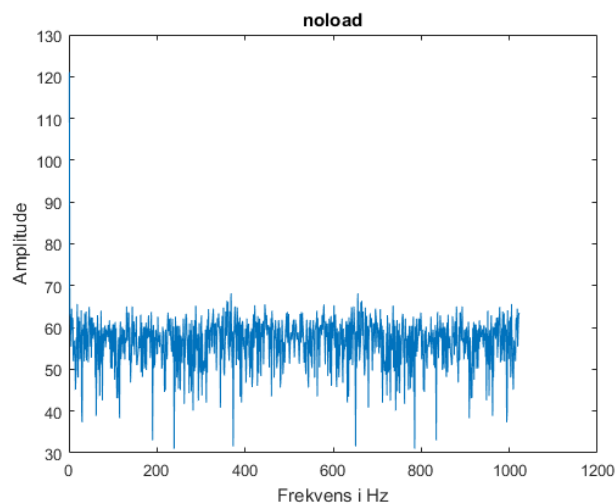
```
1 %Varians
2 noload_var = (1/size(noload,2))*sum((noload-noload_avg).^2);
3 load_var = (1/size(load,2))*sum((load-load_avg).^2);
4
5 %Spredning
6 noload_spred = sqrt(noload_var);
7 load_spred = sqrt(load_var);
```

Dette passer fint overens med histogrammerne på figur 5 og 6.

Hvis vi plotter load og noload i effekt-spektret kan man se at det ligner hvid støj, dog med en enkelt spike ved DC.



Figur 2: Effekt-spektrum for noload



Figur 3: Effekt-spektrum for load

For at finde afstanden mellem hver bit-niveau i gram, laver vi følgende udregning.

```
8 LSB_value = 1000/valueDiff
```

Dette giver os en LSB værdi på 3.3.

3 Opgave 2 - Design af midlingsfilter

Det kommende afsnit kommer til at omhandle hvordan midlingsfilteret virker og hvordan det er implementeret. Filteret er blevet testet på dataen fra opgave 1. Vi har eksperimenteret

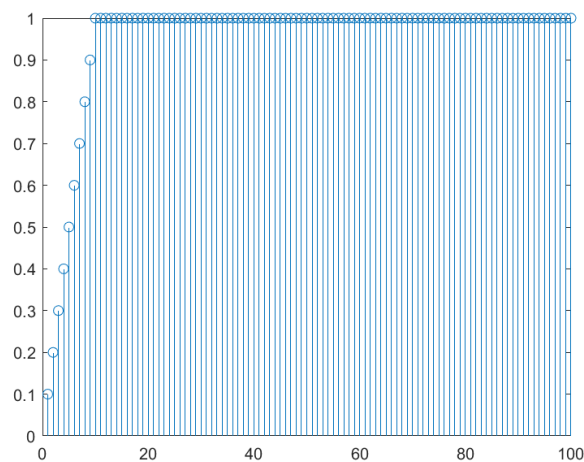
med både lineært og eksponentielt midlingsfilter.

3.1 Lineært midlingsfilter

Det lineære midlingsfilter fungerer ved løbende at tage M inputværdier og dividerer dem med M . Differensligningen for dette ser sådan ud:

$$y(n) = (1/M)(x(n) + x(n-1) + \dots + x(n-M+1))$$

Der er en indsvingningstid for filteret, da de første M værdier ikke passer overens med signalet. Det vil sige at de første M samples i filter outputtet bliver lavere. Dette kaldes indsvingningstiden. Dette kan ses på figur 4.



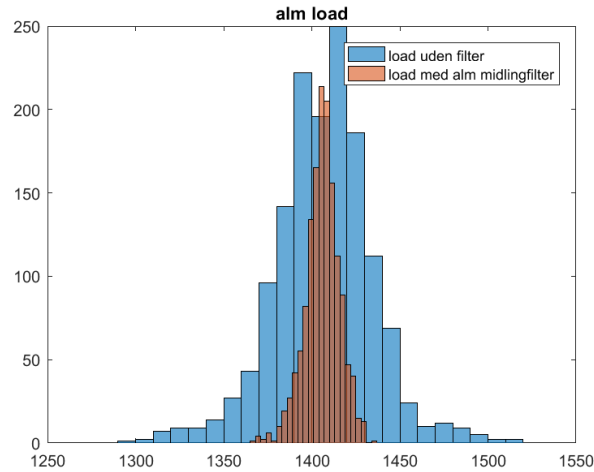
Figur 4: Histogram af load med og uden filter. Filterorden = 10

For at midle data fra opgave 1, lavede vi først et lineært midlingsfilter. Dette er gjort på følgende måde:

```
9 M = 10;  
10 h1 = 1/M * ones(1, M);
```

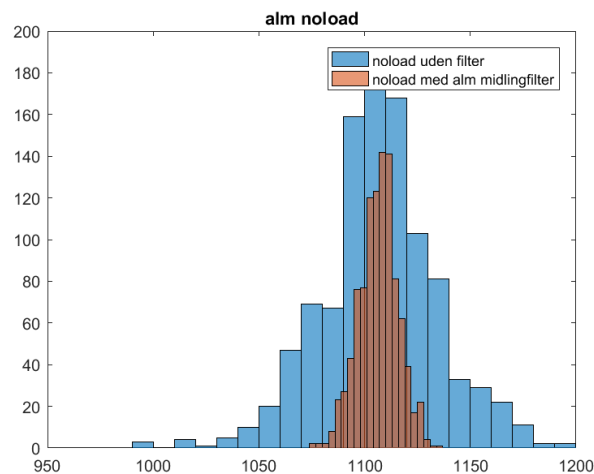
Dette laver et midlingsfilter med M koeficienter, der hver har en værdi på $1/M$. Ved at indstille på ordnen (altså M) vil man kunne få filteret til midle kraftigere.

Da dataen fra opgave 1 viser at der både er en load og noload, er filteret påført de to stadier hver for sig. Dette er blevet plottet på et histogram, som kan ses på figur 5



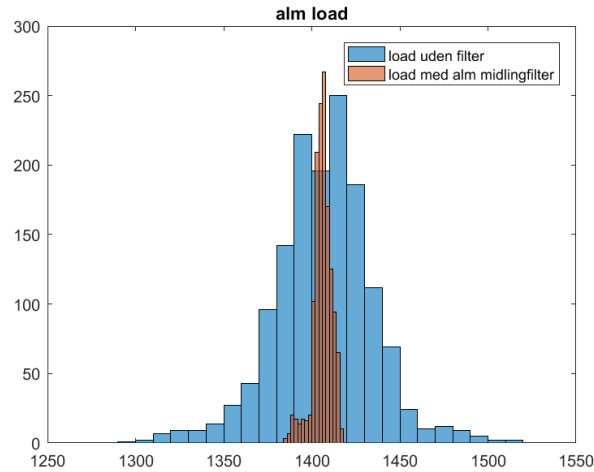
Figur 5: Filter respons for det lineære midlingsfilter

Man kan se at filteret får vores data til at ligge tættere på hinanden, hvilket er det vi gerne vil se. Det samme kan ses for noload på [6](#)



Figur 6: Histogram af noload med og uden filter. Filterorden = 10

Ved at ændre på ordnen kan man få filtret til at midle kraftigere. Dette kan ses på figur [7](#)



Figur 7: Histogram af load med og uden filter. Filterorden = 50

Det ses tydeligt at de orange pinde er blevet smallere og fylder et mindre område på grafen.

Hvis man kigger på varians og spredning kan man se at de stemmer godt overens med vores grafer.

```

11
12 var_load = var(y_load)
13 var_noload = var(y_noload)
14
15 P_load = (1/sqrt(M)*S_load)^2
16 P_noload = (1/sqrt(M)*S_noload)^2

```

Ved $M = 10$ får vi en varians på ca 90, mod en varians på ca 25 ved en orden på 50. Hvis man kigger på støj-effekten ser man det samme. Ved en orden på 10 er støj-effekten ca 10 højere end ved en orden på 50.

Hvis man ønsker en max svingningstid for sit system, bliver man muligvis nødt til at begrænse sin orden på filteret. Hvis vi har et krav om en indsvingningstid på 100ms kan vi lave en hurtigt udregning:

```

17 maxKoeff = fs*0.1;

```

Dette giver os et filter med en orden på 30.

3.2 Eksponentielt midlingsfilter

Vi forsøgte også at lege med et eksponentielt midlingsfilter. Dette midlingsfilter bruger et simpelt IIR lavpasfilter beskrevet af differensligningen:

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

Denne type filter er ofte brugt pga dens fordele, som fx at den kræver færre beregninger per output-sample end et standard midlingsfilter, og dens stærkt reduceret hukommelses brug, idet den kun indeholder et forsinkelse element $y(n - 1)$, som skal gemmes i en hukommelses-plads i fx. et embedded system. De nyeste samples $x(n)$ får også størst betydning i outputtet $y(n)$, hvilket betyder at filteret reagerer hurtigere på ændringer i input $x(n)$ et standard midlingsfilter.

α er en midlingsfaktor mellem 0 og 1 som bestemmer støjreduktionen i forhold til responstiden. Vi prøvede med en lille værdi af α i vores midlingsfilter, hvilket gav en god støjreduktionen men langsom responstid, omvendt prøvede vi også med en høj værdi af α i vores midlingsfilter, hvilket gav en mindre støjreduktionen, men hurtigere responstid. Så præcis som med det alm midlingsfilter er der et trade off mellem støjreduktionen og responstid.

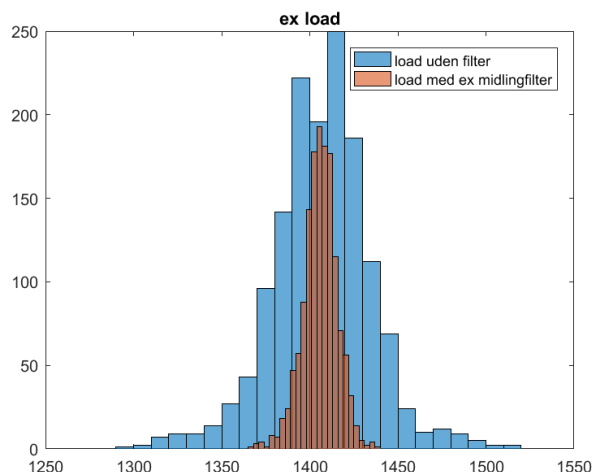
α bestemmes ved:

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

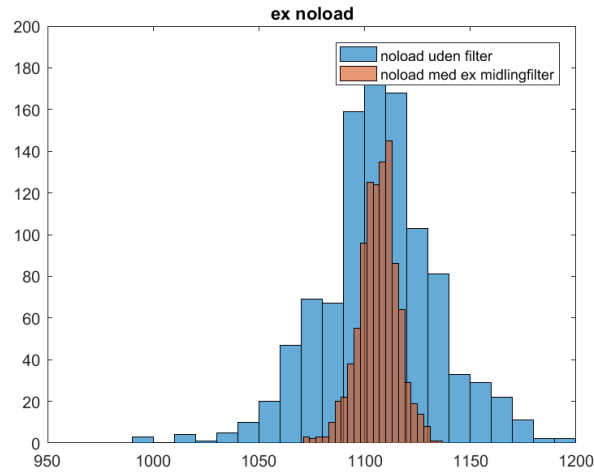
Hvor R er den faktor som støj variansen bliver reduceret. Vi kan sammenligne den eksponentiele midlingsfilters reduktion i støjeffekten direkte med et linært midlingsfilters reduktion ved at opskrive:

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

Hvor N er antal af FIR midlings koeficienter, når $N > 3$. Dette har vi prøvet at vise midling for et eksponentielt midlingsfilter med $N = 10$ stemmer nogenlunde med midlingen fra et $N = 10$ point linært midlingsfilter fra før. De nye midlinger eksponentielt midlingsfiltere kan ses i [Figure 8](#) og [Figure 9](#)

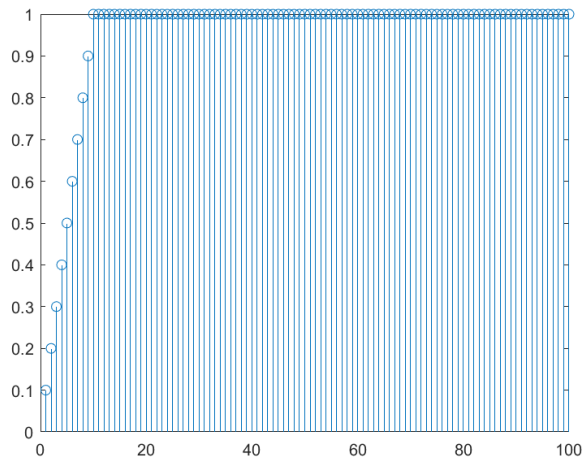


Figur 8: Histogram af load med og uden eksponentielt filter. $N = 10$

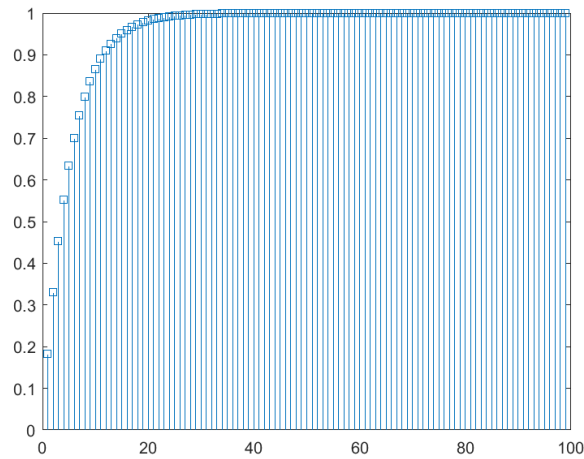


Figur 9: Histogram af noload med og uden eksponentielt filter. $N = 10$

Som man kan se minder midlingen for et eksponentielt midlingsfilter i [Figure 8](#) og [Figure 9](#) med midlingen fra de lineære midlingsfilter fra før i [Figure 5](#) og [Figure 6](#). Den store forskel er deres respons på fx. et step som kan ses i [Figure 10](#) og [Figure 11](#).



Figur 10: Linært midlingsfilter respons på step ved $M=10$.



Figur 11: Eksponentielt midlingsfilter respons på step ved $N=10$.

Som det kan ses i [Figure 10](#) og [Figure 11](#), Så er reagerer det eksponentielle midlingsfilter hurtigere på steppet end det lineære midlingsfilter, dog indhentes forspringet fra det eksponentielle midlingsfilter, idet det aldrig kan ramme slutværdien præcis, men blot komme uendelig tæt på. Alligevel tager det ca dobbelt så mange samples for det eksponentielle midlingsfilter at nå til næsten slutværdiden i forhold til det lineære filter som når slutværdien efter dens 10 koefficienter er kørt igemme signalet. Dette skyldes at eksponentielle midlingsfilter er et IIR filter som jo har uendeligt respons på et signal.

4 Opgave 3 - System overvejelser

5 Matlab kode