

# Case Project 4 – Sonar

Navn	Studienummer
Naiyun Wu	201405716
Frederik Gadegaard	201405625
Maciej Lech	20107672
Jacob Aagaard	201404442

## Indhold

Intro.....	3
Opgave 1 - Signal generation .....	3
Opgave 2 - Implementering af lyd-afspilning/-optagelse.....	5
Opgave 3 - Eksperimenter .....	7
Opgave 4 - Signal analyse i Matlab .....	7

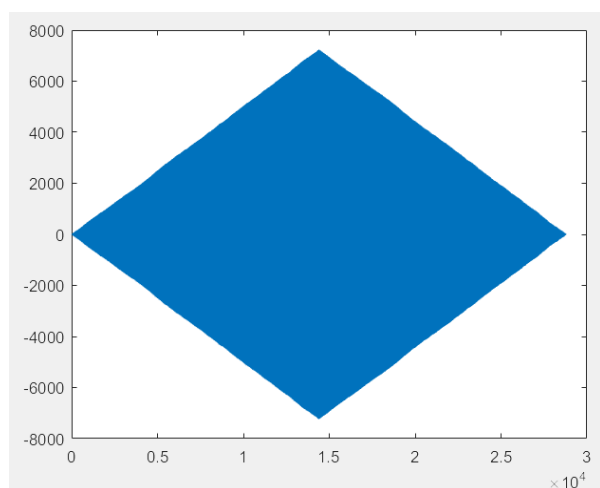
## Intro

I denne opgave skal vi implementere et system som kan måle afstanden til et objekt vha. krydskorrelation. Vi vil anvende matlab til dannelse af signalet som skal udsendes og derefter anvendes i krydskorrelationen, og så vil vi anvende Blackfin-kittet til selve afspille/optage funktionaliteten.

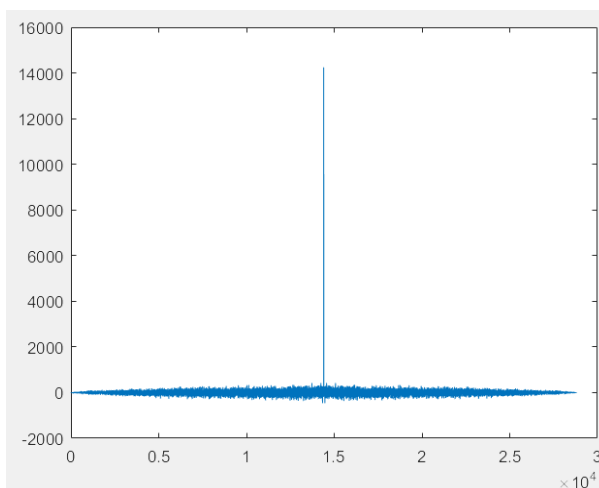
## Opgave 1 - Signal generation

Når vi skal overveje hvilket signal der er det bedste at bruge til at sende og optage, er det vigtigt at kigge på hvordan forskellige signaler opfører sig, når de bliver krydskorreleret med hinanden.

Da en sinus ligner sig selv meget, er det svært at se hvor toppen er. På Figur 1, har vi krydskorreleret en ren sinus med sig selv.



Figur 1 Krydskorrelation af sinus med sig selv



Figur 2 Krydskorrelation af chirp med sig selv

Det kan ses at toppen er i midten som forventet, men det er meget sværere end hvis man anvendte et hvidstøjssignal eller en chirp. Figur 2 viser hvad der sker når en chirp bliver krydskorreleret med sig selv. Her ses det, at når krydskorrelationen for et hvidstøjssignal eller en chirp rammer lidt ved siden af, så er resultatet stadig tæt på 0. Det er derfor meget nemt for disse signaler at aflæse, hvornår de ligger fuldstændig oveni hinanden, under krydskorrelationen. Fordelen ved et sinus signal er, at det er nemt at implementere.

Udregningen af, hvad en sample svarer til ses nedenfor:

$$f_s := 48000 \text{ Hz} \quad T := \frac{1}{f_s} = (2.083 \cdot 10^{-5}) \text{ s}$$

$$v_{\text{sound}} := 340.29 \frac{\text{m}}{\text{s}}$$

$$T \cdot v_{\text{sound}} = 7.089 \text{ mm}$$

Figur 3 Længden af lydsignalet i mm

Ud fra dette ses, at længden af en sample er omkring 7mm, hvilket betyder at hver gang vi udsender en sample, kan den sample rejse 7mm væk fra højttaler før den næste sample bliver sendt afsted.

Længden af signalet vi udsender er afhængig af minimumsafstanden vi gerne vil måle. Vi skal helst ikke starte med at optage lyden før afspilningen færdiggøres, fordi vi da kan risikere at krydskorrelere det udsendte signal. Vi har defineret en minimumsafstand på 1 meter. Tiden det tager for lydsignalet til at rejse frem og tilbage vises herunder:

$$v_{\text{sound}} := 340.29 \frac{\text{m}}{\text{s}} \quad d := 1 \text{ m}$$

$$\frac{d \cdot 2}{v_{\text{sound}}} = 5.877 \text{ ms}$$

Figur 4 Varighed af lydssignal i sekunder

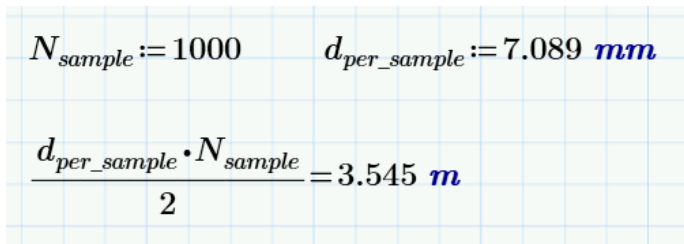
Det betyder at længden af lydsignalet vi kan udsende skal være mindre end 5.8ms med en minimum målingsafstand på 1 meter.

Man kan sige, at der et tradeoff mellem længden af lyden og minimumsafstanden. Jo længere væk objektet er, jo flere samples kan vi udsende, samt minimumsafstanden øges og vice versa.

Den maximale afstand vi kan måle er afhængig af amplituden af det udsendte signal. Signalet vil blive dæmpet igennem luften, sådan at det reflekterede signal vil være svagere end det udsendte signal. Den maximale afstand vil være der hvor det reflekterede signal er så svagt, at vi ikke længere kan detektere det.

En anden begrænsning er længden af optagelses array. Da blackfinkittet har kun 10-20kB af L1 hukommelse, så er størrelsen af arrayet vi ligger den optaget lyd begrænset.

For dette opgave, har vi defineret det til at være 1000. Med 1000 plads i array, så er max afstand vi kan måle lige under


$$N_{sample} := 1000 \quad d_{per\_sample} := 7.089 \text{ mm}$$
$$\frac{d_{per\_sample} \cdot N_{sample}}{2} = 3.545 \text{ m}$$

Figur 5 Længste afstand vi kan måle med  $N = 1000$  samples

Der skal deles med 2 fordi det udsendt signal skal rejse frem og tilbage. Objekt længere end 3.5m kan ikke optages fordi arrayet er ikke stort nok.

## Opgave 2 - Implementering af lyd-afspilning/-optagelse

Vi valgte at benytte blackfin-kittet til afspilning og optagelse. Vi anvendte her den vedlagte funktionalitet til konverteringen af et array af decimaler til et array af hexadecimaler.

I første omgang ville vi blot afspille en lyd - længere end chirpen - på blackfin for at tjekke om den del af koden virkede. Her opstod dog en kompileringsfejl, hvilket vi fandt ud af var fordi lydsignalet var for langt i forhold til den allokerede hukommelse i blackfin-kittet.

Vi har implementeret vores lyd-afspilning/-optagelse sådan, at når vi trykker på SW4 på blackfin-kittet afspilles signalet vha. højttaleren, imens det optages med mikrofonen. Dette vil også resultere i at vi optager lyden der afspilles før den reflekteres tilbage fra objektet, hvilket vil være synligt når vi krydskorrelerer de to signaler.

Dette udnytter vi i vores plot af krydskorrelationen, hvor vi sammenligner peakene for hhv. det udsendte chirp-signal og det returnerede chirp-signal.

Herunder ses kodeudsnittet for Process\_data, som er opdelt af 2 switch-cases, PLAY\_AND\_RECORD og WAIT\_STATE. I den første state afspilles og optages lyden og den anden case er blot til at hoppe ned i efter hele optagelsen er gennemført, sådan at det ikke afspiller og optager konstant.

```

void Process_Data(void)
{
    // FlagAMode is changed by using pushbutton SW4 on board..
    switch (FlagAMode) {
        case PLAY_AND_RECORD :
            // checks if all of the chirp-signal has been played
            if(i < 241)
            {
                //Chirp output from channel0 (one of these channels (0 or 1) is redundant)
                iChannel0LeftOut = (((int)mysig[i])<<14);
                iChannel0RightOut = (((int)mysig[i])<<14);

                //Chirp output from channel1 (one of these channels (0 or 1) is redundant)
                iChannel1LeftOut = (((int)mysig[i])<<14);
                iChannel1RightOut = (((int)mysig[i])<<14);
            }

            // Recording the first part of signal (first 241 samples)
            recording[i] = iChannel1RightIn;

            // Incrementing i
            i++;

            // checks if the desired recording length (1000 samples) has been reached
            if(i==5000)
            {
                FlagAMode = WAIT_STATE;
                i = 0;
            }
            break;

            // this case is just for waiting, so that we are not playing/recording all the time
        case WAIT_STATE :
            break;
    }
}

```

Figur 6 Kodeudsnit for Process\_data.c

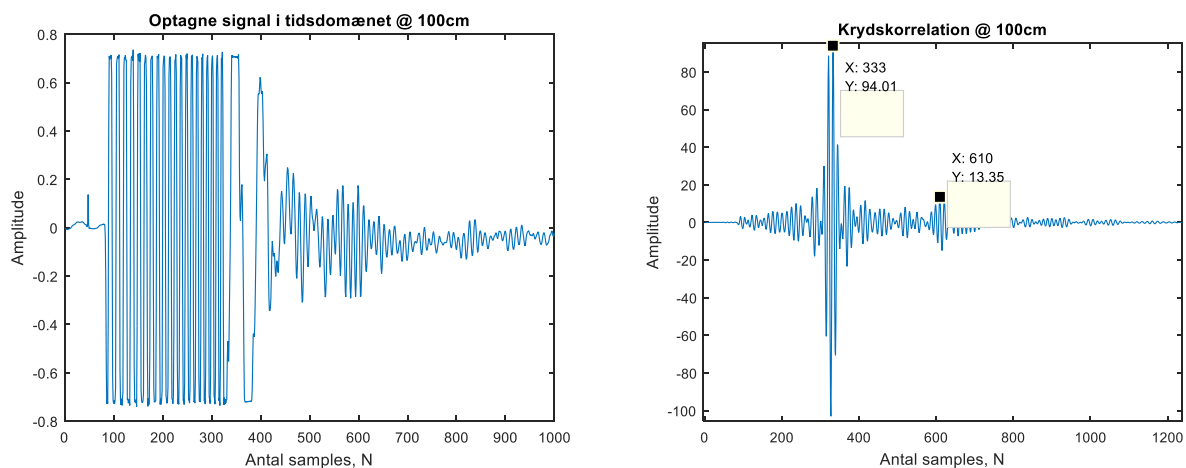
## Opgave 3 - Eksperimenter

Vi har lavet forskellige målinger ved forskellige afstande. Se opgave 4.

## Opgave 4 - Signal analyse i Matlab

Nedenfor har vi plottet de optagne signaler i Matlab i tidsdomænet, for at verificere at vi har optaget noget der minder om en chirp.

Fra vores første måling ser vi følgende signal, når vi peger højttaler/mikrofon mod en mur i en afstand af ~100 cm. Det betyder dermed at lyden rejser omtrent 200cm fra højttaleren indtil det rammer mikrofonen igen.

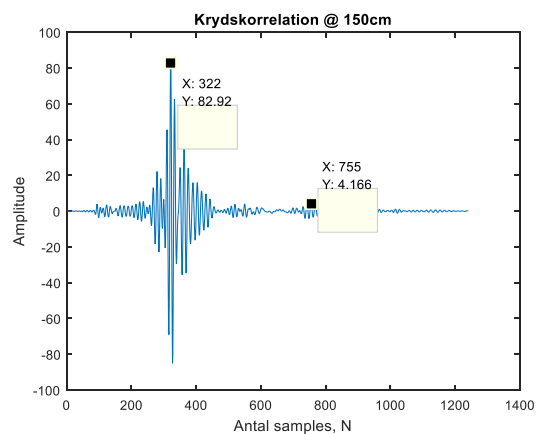
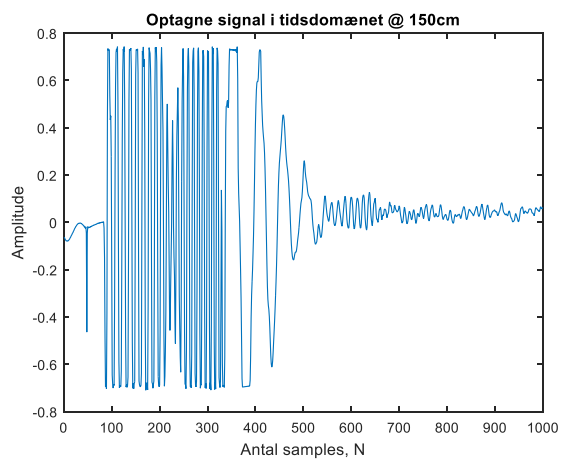


Figur 7 Afstand = 100cm

Den første peak i krydskorrelationen viser det udsendte signal, og peak nummer 2 viser det optagne signal. For at omregne antal samples i krydskorrelationen til den målte afstand i centimeter, laver vi følgende udregning:

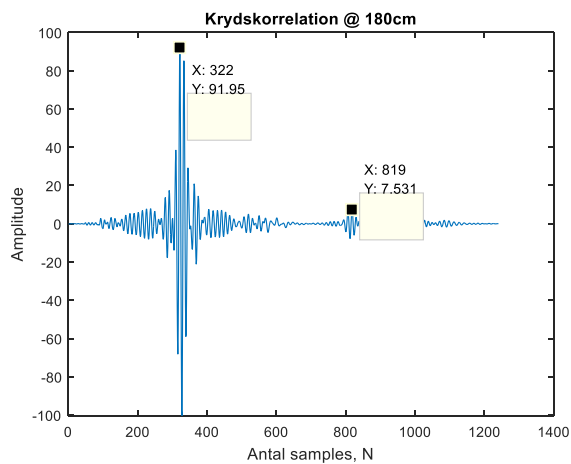
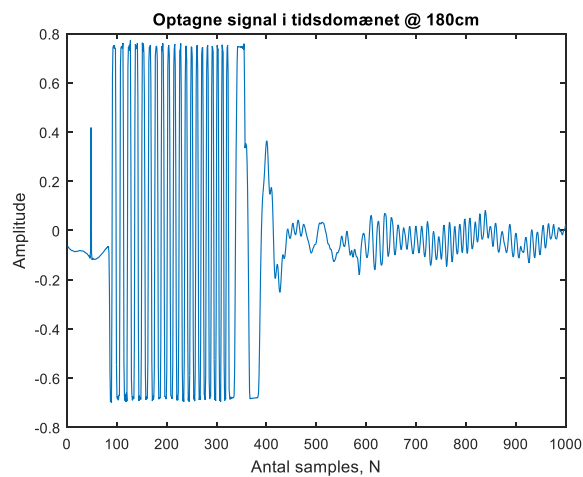
$$afstand_{100} := (610 - 333) \cdot 7 \text{ mm} = 193.9 \text{ cm}$$

Denne metode går naturligvis igen i følgende figurer:



Figur 8 Afstand = 150cm

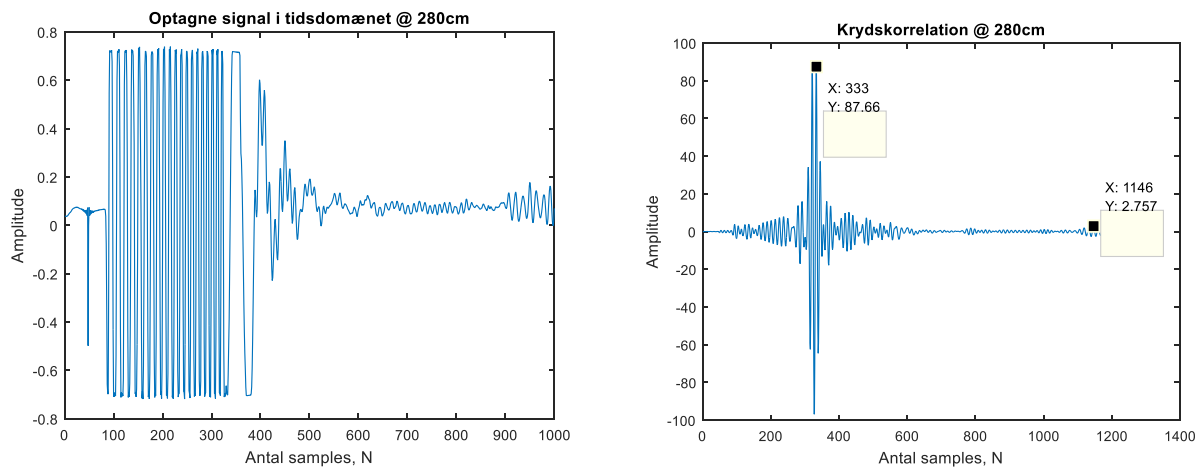
$$afstand_{150} := (755 - 322) \cdot 7 \text{ mm} = 303.1 \text{ cm}$$



Figur 9 Afstand = 180cm

$$afstand_{180} := (819 - 322) \cdot 7 \text{ mm} = 347.9 \text{ cm}$$





Figur 10 Afstand = 280cm

$$afstand_{280} := (1146 - 333) \cdot 7 \text{ mm} = 569.1 \text{ cm}$$

Vores sonar målinger er opstillet i en tabel nedenfor, hvor måle-afvigelsen er opstillet sammen med den teoretiske værdi:

Teoretisk værdi (cm)	Målt værdi (cm)	Afvigelse (%)
200	193.9	3.05
300	303.1	1.033
360	347.9	3.361
560	569.1	1.625

Vi har implementeret vores egen funktion til krydskorrelation på følgende måde:

```
function y = myccconv(x1,x2)

n1=length(x1);
n2=length(x2);

x2=flip1r(x2); %Flip signal for x-correlation.

yy=zeros(1,n1+n2); %Initialize output-array with zeros.

for n=1:n1+n2 %Run through length of both signals combined.
    for m=1:n1
        if n-m < 1 || n-m > n2
            yy(n)=yy(n)+0;
        else
            yy(n)=yy(n)+(x1(m).*x2(n-m));
        end
    end
end

y=yy;
end
```

Figur 11 Vores egen funktion til krydskorrelation