I3DSB - Mini projekt 1

Digitale bølger

SW	Rasmus Møller Nielsen au633064

Hold nr. 13

1 Introduktion

2 Opgave 1 - Find antal samples

Som udgangspunkt har vi fået udleveret 3 filer, hvor 2 af dem (S_1 og S_3) er mono imens at S_2 er stereo. Disse filer skal indlæses i matlab og gemmes som en lang række af samples.

Dette gør man ved hjælp af følgende kode:

Listing 1: Indlæsning af samples fra fil

```
1 % Loading files into arrays of samples
2 [y(1).sample, ~] = audioread('Signal_s1.wav');
3 [y(2).sample, ~] = audioread('Signal_s2.wav');
4 [y(4).sample, Fs] = audioread('Signal_s3.wav');
```

Som vi kan se i Listing 1 bbliver filerne indlæst med functionen *audioread('filename')*. Funktionen returnerer her en array af samples og den frekvens der er optaget med. Vi får at vide at alle frekvenserne er 44.100 Hz.

For herefter at finde ud af hvor mange samples der er per signal kan man bruge funktionen *length()* til at finde antallet af samples. (Ellers vil man også kunne aflæse dem ude i ens workspace).
Længden på signalerne bliver:

$$length(S_1) = 4213759$$

$$length(S_2) = 8753617$$

$$length(S_3) = 1270957$$

$$Fs = 44.100Hz$$

Antallet af samples vil senere blive refereret til som nS (number of Samples).

3 Opgave 2 - Plot signaler

For at plotte signalerne bruger man funktionen *plot(x, y)*. Hertil kan man bruge funktionerne *xlabel("label")*, *ylabel("label")* og *title("title")* til at tilføjer titler til akserne og hele plottet.

Da jeg opbevarer mine signaler og deres værdier i en struct bruger jeg en for løkke til at tilgå dem. Før jeg kan plotte skal jeg udregne tiden i sekunder som signalet løber over. Dertil bruger jeg formlen:

$$t = nS \cdot \frac{1}{Fs}$$

Her er *Fs* frekvensen for sampling. Så $\frac{1}{Fs}$ vil være tiden per sampel og ved at gange det med antallet af sampels får man tiden for hele signalet.

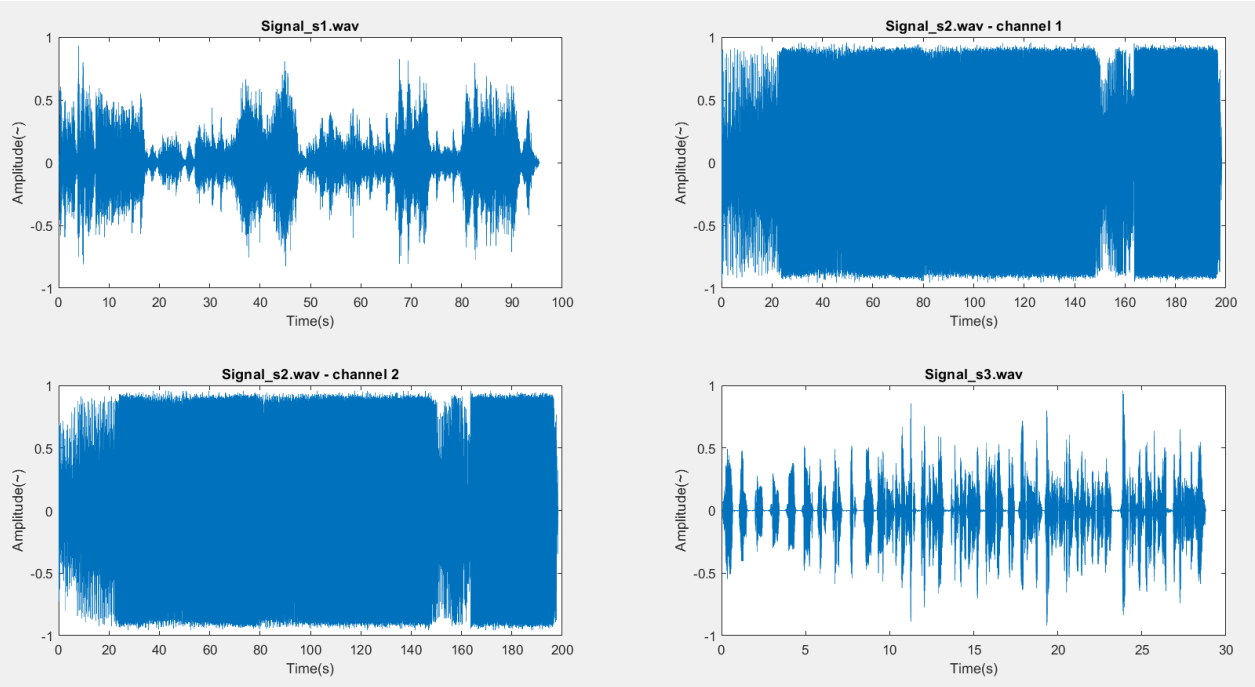

Figure 1: Plot af signaler med korrekte akser og titel

På Figure 1 kan man se begge kanaler af S_2 samt S_1 og S_3 alle med Tiden t på X-aksen, Amplitude på Y-aksen og en passende titel til hvad det viser.

4 Opgave 3 - Find værdier for signalerne

I det her afsnit skal jeg finde følgende værdier for signalerne

- max
- min
- mean (gennemsnit)
- rms
- effekt

Til de første 4 vil jeg bruge følgende funktioner:

- *max()*
- *min()*
- *mean()*
- *rms()*

og for at udregne effekten vil jeg bruge følgende formel:

$$E = \sum {A_{samples}}^2$$

Efter at have kørt funktionerne og brugt formlen til udregning af effekt får jeg følgende værdier:

- S_1
  - max = 0.9306
  - min = -0.8205
  - mean = $-4.28 \cdot 10^{-5}$
  - rms = 0.1199
  - effekt = $6.0558 \cdot 10^4$
- $S_2 Venstre$
  - max = 0.9558
  - min = -0.9558
  - mean = $-1.94 \cdot 10^{-5}$
  - rms = 0.3383
  - effekt = $1.0018 \cdot 10^6$
- $S_2 Højre$
  - max = 0.9558
  - min = -0.9558
  - mean = $-1.94 \cdot 10^{-5}$
  - rms = 0.3548
  - effekt = $1.1022 \cdot 10^6$
- S_3
  - max = 0.9600
  - min = -0.9192
  - mean = $-5.74 \cdot 10^{-4}$
  - rms = 0.0919
  - effekt = $1.0739 \cdot 10^4$

5 Opgave 4 - Find crest faktorer og sammenlign

Til udregning af Crest faktoren bruger vi følgende formel:

$$C = 20 \cdot log_{10}(\frac{max}{rms})$$

$$C_{S_1} = 17.8006$$

$$C_{S_2 venstre} = 9.0213$$

$$C_{S_2 højre} = 8.6064$$

$$C_{S_3} = 20.3770$$

6 Opgave 5 - Nedsample signal 1 med faktor 4

I følgende opgave skal jeg tage S_1 og nedsample det med en faktor 4. Hertil bruger jeg funktionen *decimate()*, hvor argumenterne er listen af samples fra S_1 og derefter den faktor som det skal nedsamples med.

Listing 2: Nedsampel med faktor 4

```
1 y(5).sample = decimate(y(1).sample, 4);
```

Hvis man herefter plotter det får man som på Figure 2

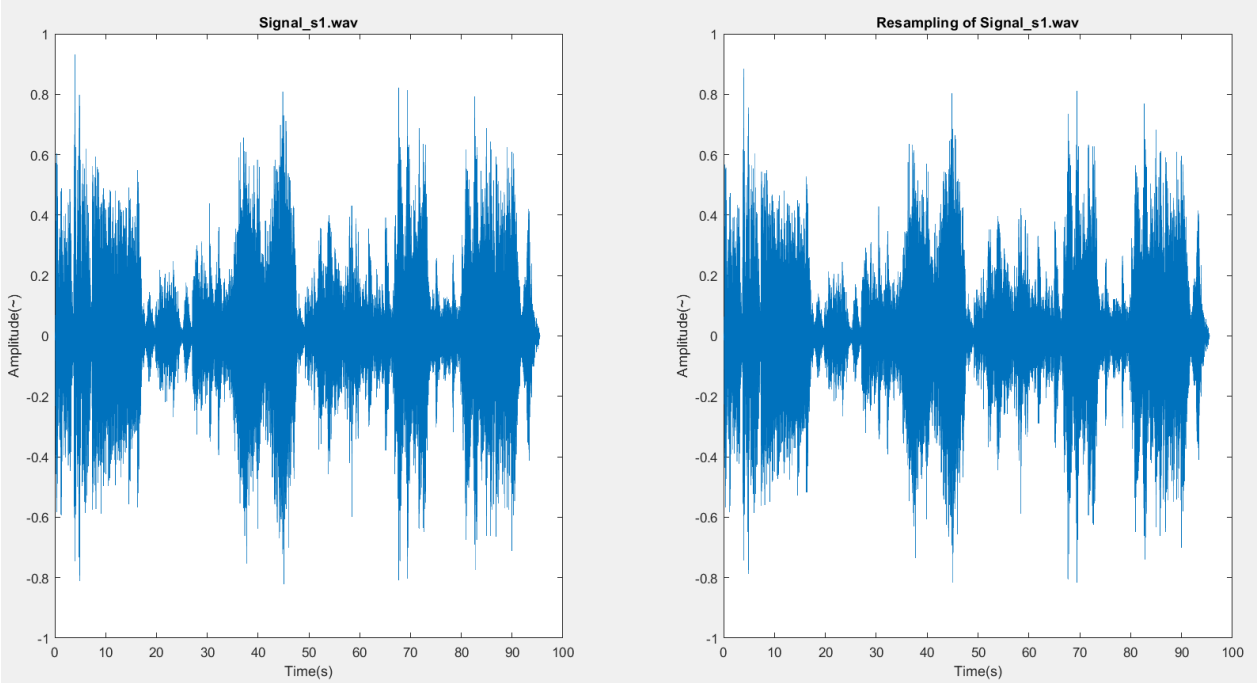

Figure 2: Downsampling af signal 1

7 Opgave 6 - Beskriv forskellen på signalet fra opgave 5

Udfra plottet i Figure 2 kan man se der er nogle små ændringer, specifikt lige omkring 70 sekunder, hvor man kan se, at det resamplede signal ikke får samme højde som det originale. Derudover kan man også høre, at selvom signalet holder sin integritet, mister den en stor del af dybden og kvaliteten af lyden.

8 Opgave 7 - Sammenlign de to kanaler i signal 2

I denne opgave skal man sammenligne de to kanaler i S_2 og beskrive forskelle og ligheder.

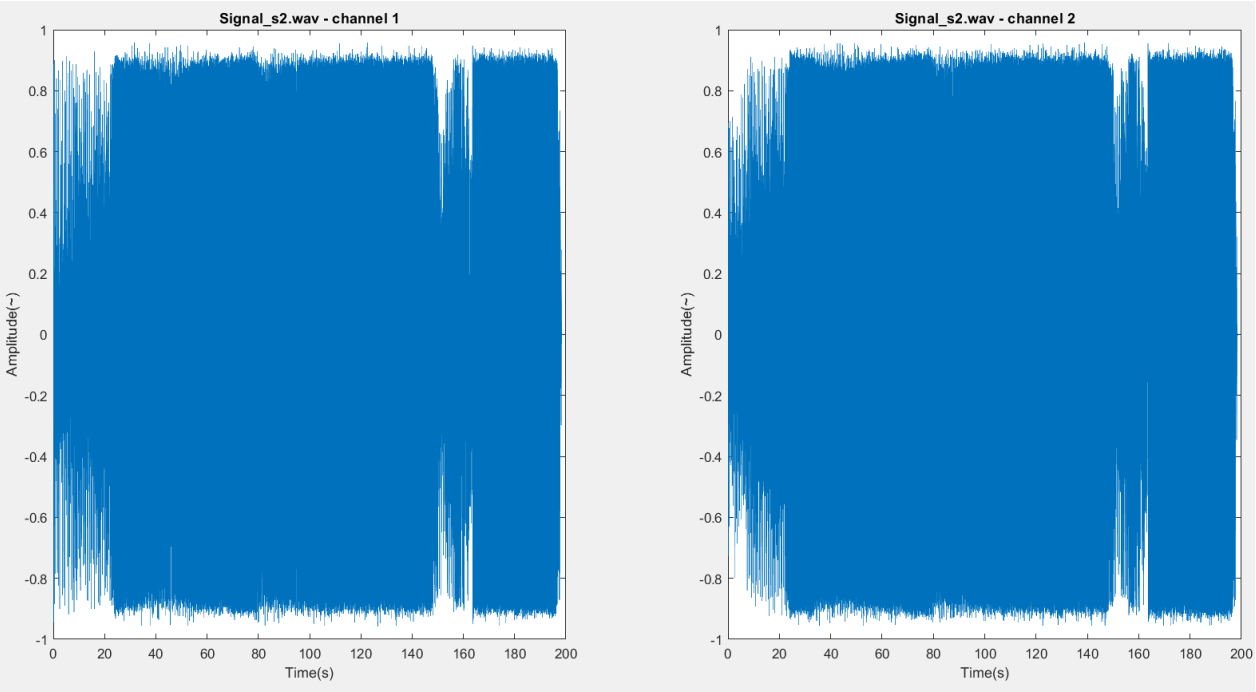

Figure 3: Begge kanaler af Signal 2

På Figure 3 kan man se, at kanal 1 starter højt, hvorimod kanal 2 starter lavt og så kommer op på samme amplitude som kanal 1 når man kommer lidt længere ind i signalet.

9 Opgave 8 - Kvantiser signal 2(venstre) med 4 bit

I denne opgave skal man tage venstre side af signal 2 og kvantisere det til 4 bit. Hertil har jeg brugt den funktion vi har fået givet *quantizeN()*. Denne funktion tager en liste af samples og så den mængde bits som man vil kvavntisere den til. Efter at kvantisere til 4 bit kan man plotte det og få resultatet som ses på Figure 4

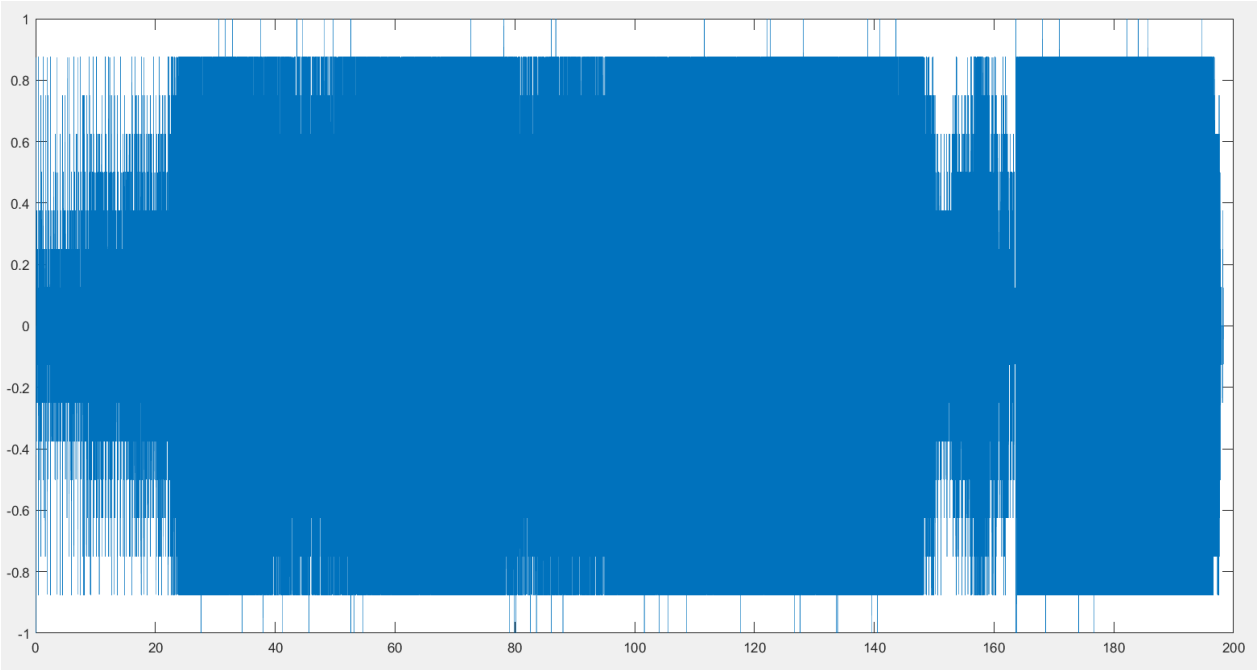

Figure 4: 4 bit kvantisering af signal 2's venstre kanal

.

Til sammenligning med Signal 2's venstre kanal på Figure 1 kan man se, at der er meget mere detalje på det originale signal. Dette giver god mening efter vi har gjort det kvantiserede signal's bredde til 4 bit. Derudover når man afspiller det kvantiserede signal lyder det bare som radio støj.

10 Opgave 9 - Lav fade out på signal 3

I denne opgave skal man lave et eksponentielt fald til 5% over den sidste tredjedel af signal 3.

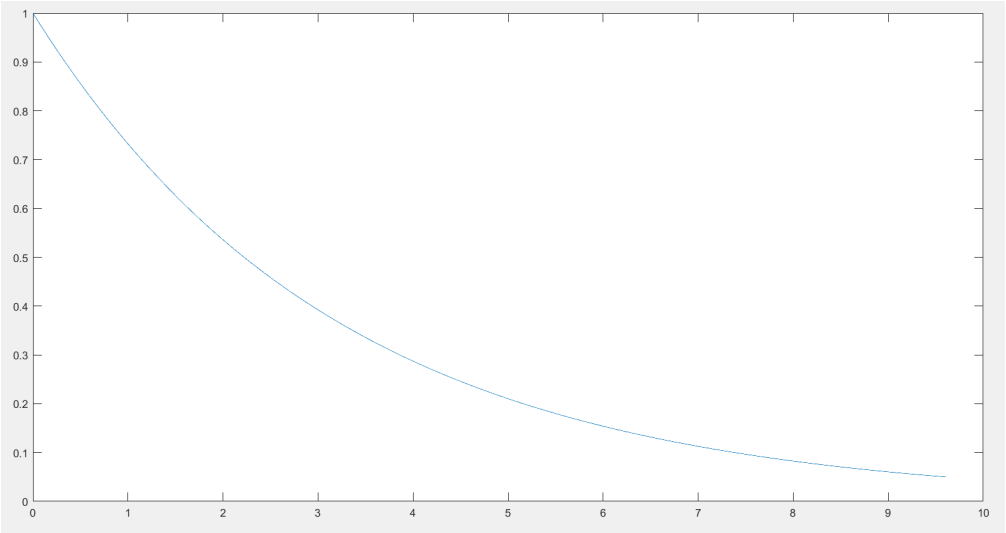

Figure 5: Negativ eksponentiel funktion

På Figure 5 kan man se, hvordan vores eksponentielle funktion skal se ud. For at gøre dette skal man først have en funktion som laver dette fald fra 100% til 5%. Til dette bruger jeg følgende formel:

$$f(x) = b \cdot a^x$$

Da vi har et fald fra 100% ved vi at b skal være 1. Dermed kan vi forkorte formlen til følgende:

$$f(x) = a^x$$

Herefter skal vi ha fundet vores a og dette kan man gøre ud fra følgende formel:

$$a = \frac{y_2}{y_1}^{\frac{1}{x_2 - x_1}}$$

Her skal vi indsætte vores tidsrum $x_2 - x_1$ og vores breddefald $\frac{y_2}{y_1}$.
Fra opgaven ved vi at tidsrummet er fra sidste tredjedel til slutningen, samt at breddefaldet går fra 100% til 5%. Så med følgende kode kan man lave en eksponentiel værdi som man bagefter vil kunne gange på den sidste tredjedel af vores samples fra signal 3.

Listing 3: Oprettelse af eksponentielt aftagende funktion

```
1  t = y(4).time(1:y(4).nS/3+1);
2  a = (0.05/1)^(1/(t(end)-t(1)));
3  e = a.^t;
```

Herefter har vi fået e som er en array på størrelsen af den sidste tredjedel af signal 3 og har procentværdierne som man kan gange på vores originale signal.

Listing 4: Oprettelse af eksponentielt aftagende funktion

```
1  y(7).sample(end-length(t):end) = e.* y(7).sample(end-length(t):end);
```

Så ved at køre koden fra Listing 4 kan man få en eksponentiel aftagende sidste tredjedel af signal 3. Dette kan ses på Figure 6.

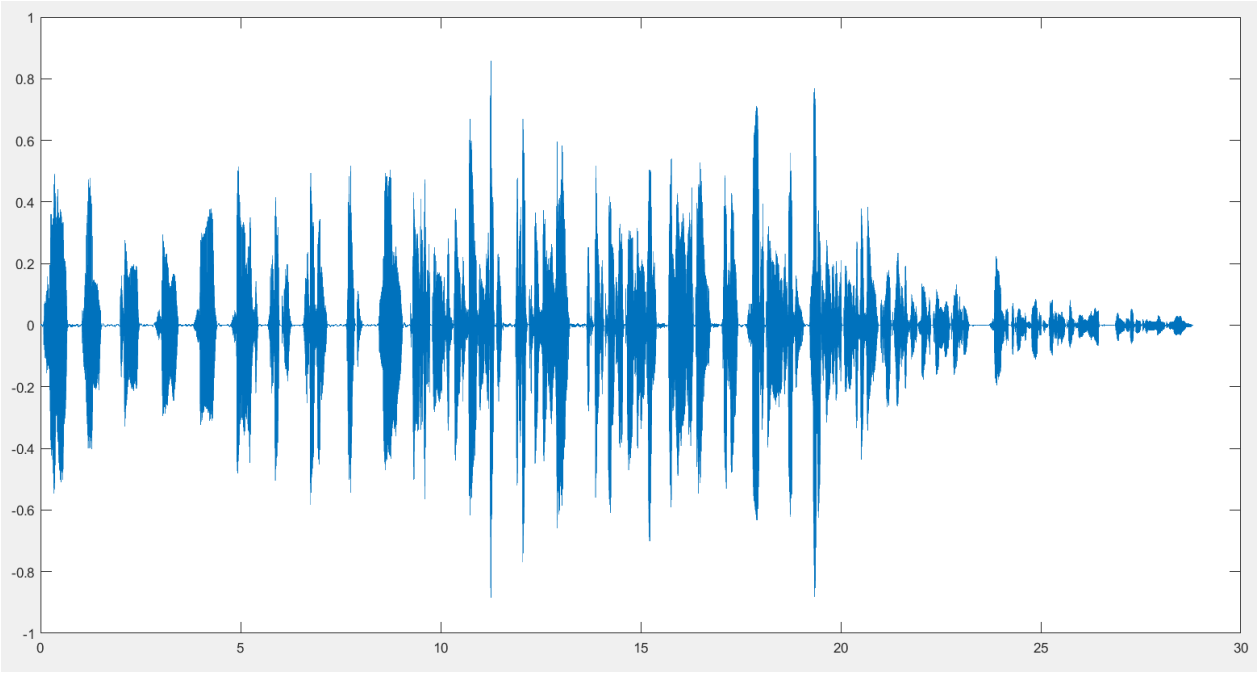

Figure 6: Fadeout på signal 3

11 Opgave 1.15 - Ekko

I denne sidste opgave handler det om, at skabe et ekko og se hvad der sker når man ligger et ekko oven i det originale ikke forskudte signal.

Først skal man lave et signal. På Listing 5 kan man se koden for at lave et signal med en frekvens på 2350 Hz og en amplitude på 3.

Listing 5: Oprettelse af ikke forskudt tone

```
Fs = 5000; %Samplingsfrekvens
Ts = 1/Fs; % Tid per sampling

length = 1; % Længde i sekunder
Ns = Fs*length; % Antallet af samples
t = [0:Ns-1]*Ts; % Tiden over samplingen

%Tone beskrivelse
f0 = 2350; %2350 Hz tone
A = 3; % Amplitude er 3
tone = A * sin(f0*2*pi*t); %Tonen som funktion
```

Denne del kode vil generere signalet som man kan se på Figure 7

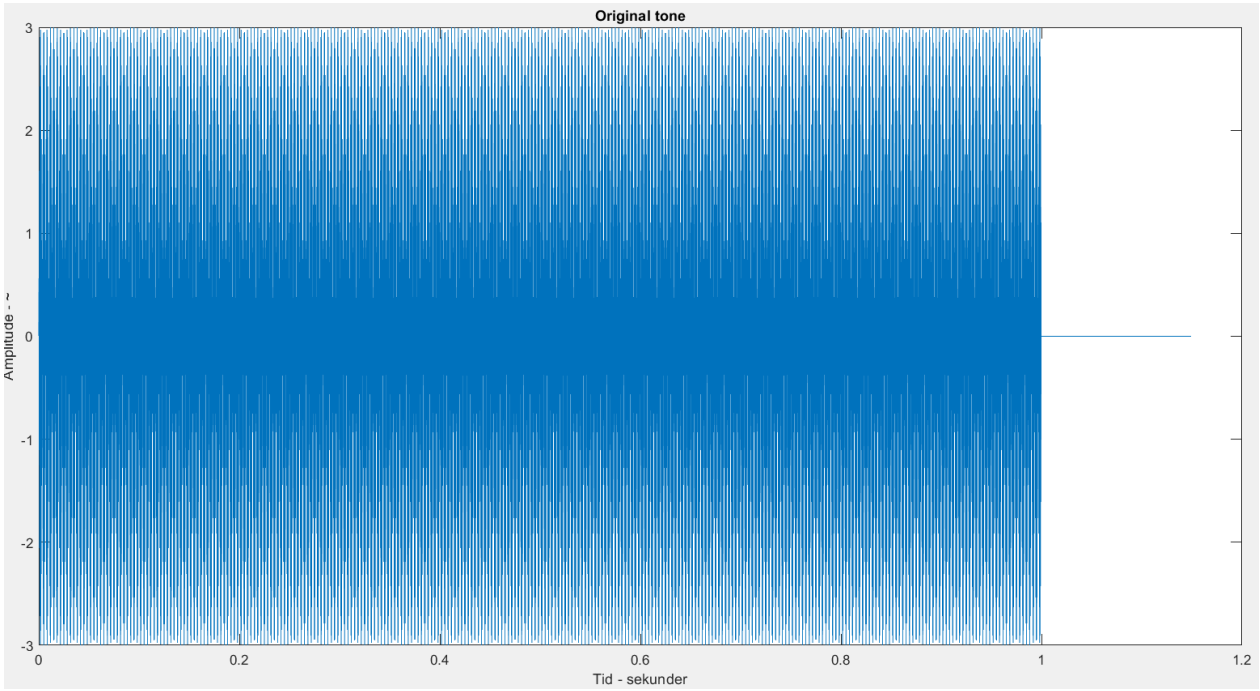

Figure 7: Original tone

Efter den er blevet oprettet skal man oprette et ekko, som egentlig bare er den originale tone forskudt. Opgaven specificere, at ekkoet originalt skal forskydes med 150 ms.

Listing 6: Oprettelse af ekko

```
1 e.delay = 150; %ekko med 150 ms forsinkelse
2 e.Ns = e.delay * (Fs/1000); %Antallet af samples i ekkoet
3
4 tone_ekko = [zeros(1,e.Ns), tone]; %Forsinker ved at smide 0'er ind foran
5 tone_ext = [tone, zeros(1,e.Ns)]; % Forlænger ved at smide 0'er ind bagved
```

I Listing 6 kan man se, hvordan vi tilpasser ekkoet og den originale tone så man nemt kan ligge dem oven i hinanden. På Figure 8 kan man se, at ved en frekvens på 2350 Hz med et ekko på 150 ms får et signal som går ud med sig selv i det tidsrum, hvor de krydser. Dette resulterer i, at man hører to korte bip med et mellemrum imellem. Herimod, hvis man ændrer ekkoet´s forskydning får man en kontinuerlig lyd, som ændres i styrke alt efter om tidsrummet er efter ekkoet er startet og før det originale er slut. Dette skyldes, at der enten sker positiv interferens (ved 300ms ekko på Figure 9) eller negativ interferens (ved 150ms på Figure 8).

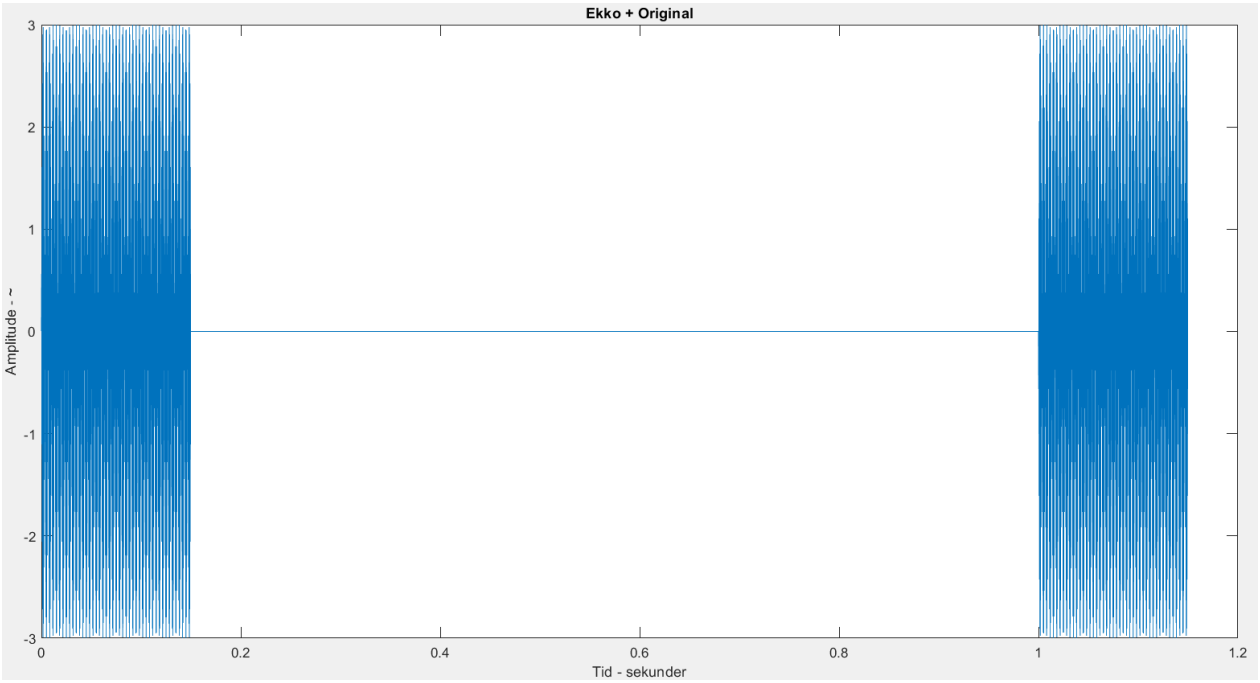

Figure 8: Ekko og original tone lagt sammen

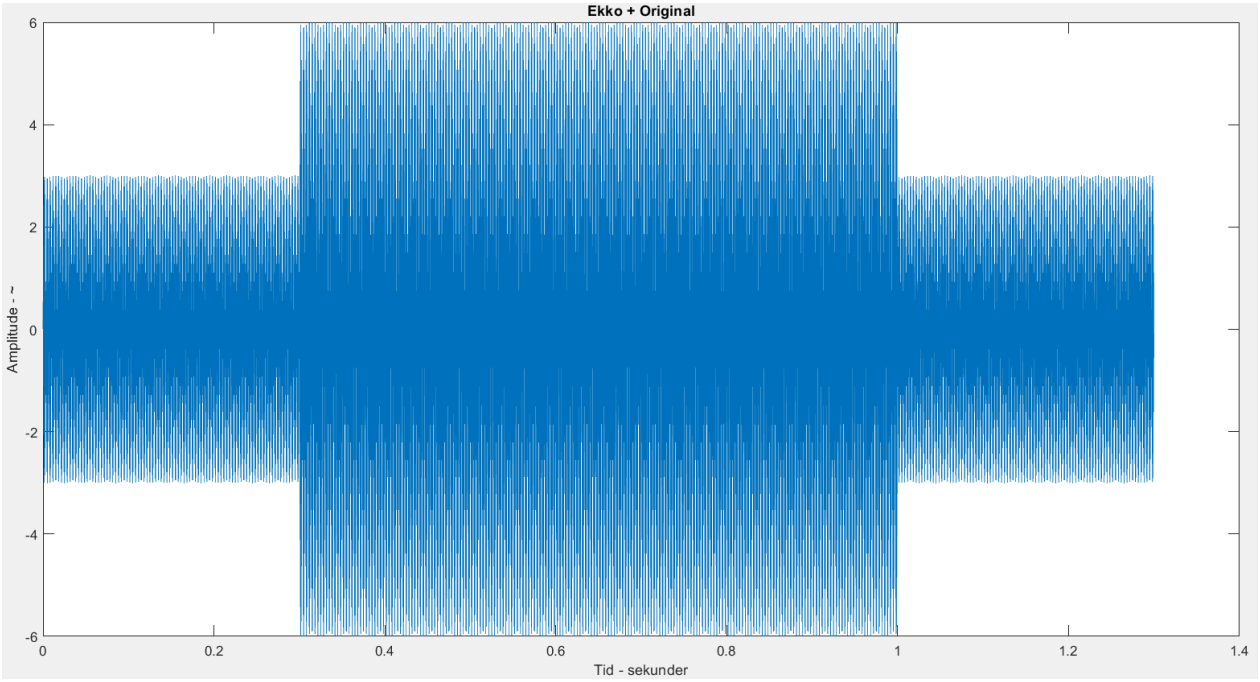

Figure 9: Ekko på 300ms