I3DSB - Mini projekt A

Sampling og analyse af digital musik- og talesignaler

SW - Gustav Nørgaard Knudsen - au612485

Hold nr. 13

1 Introduktion

Formålet med dette miniprojekt er at give en forståelse for hvordan MATLAB håndterer digitale signaler såsom lyd og hvordan disse maniuleres. Lyd opbevares i matricer i MATLAB, som så kan plottes til deres tidsenhed, for at skabe plots af lydsignalet. Desuden er det også muligt at høre lyde, f.eks. med funktioner som soundsc()

Før jeg kan begynde på opgaverne, starter jeg med at indlæse mine .wav filer med audioread(). Jeg opretter en struct, der kan indeholde alt information om mine forskellige lydfiler, samt får opdelt s2 op i left og right dele. Dette kan ses på listing 1. Desuden nulstiller jeg alle tidligere defineret variabler med med **clear** og lstinlineclc, for ikke at få overlap fra tidligere.

Listing 1: Opsætning af MATLAB

```
clear;
clc;

% Define samples
s1 = 1;
s2left = 2;
s2right = 3;
s3 = 4;

% Loading files into arrays of samples
[y(s1).sample, ~] = audioread('Signal_s1.wav');
[y(s2left).sample, ~] = audioread('Signal_s2.wav');
[y(s3).sample, Fs] = audioread('Signal_s3.wav');

% Splits signal 2 into its own channels
y(s2right).sample = y(s2left).sample(:,2);
y(s2left).sample = y(s2left).sample(:,1);

% Name samples
y(s1).name = "Signal\_s1.wav";
y(s2left).name = "Signal\_s2.wav - channel 1";
y(s2right).name = "Signal\_s2.wav - channel 2";
y(s3).name = "Signal\_s3.wav";
```

For at gøre nogle af de kommende øvelser nemmere at skrive og forstå, er har jeg oprettet et for-loop, der kan løbe igennem alle structs, og oprette de variabler og matricer der kan blive nødvendige.

```
% Predefine variables for s1, s2left, s2right, s3 and s1reS
for i = 1:length(y)
    y(i).sample = y(i).sample';
    % number of Samples 1, 2, 3
    y(i).nS = length(y(i).sample);
    %Calculating x axis for signals
    y(i).time = [0:y(i).nS-1]*(1/Fs);
    %Find max, min, mean, rms & effect
    y(i).max = max(y(i).sample);
    y(i).min = min(y(i).sample);
    y(i).mean = mean(y(i).sample);
    y(i).rms = rms(y(i).sample);
```

```
13          y(i).effect = sum(y(i).sample.^2);
14          %Calculating Crest value
15          y(i).crest = 20*log10(y(i).max/y(i).rms);
16      end
```

2 Opgave 1

For at finde antallet af samples for alle 3 signaler, kan jeg ganske simpelt kalde funktionen **length**() på alle samples. **length**() returnerer antallet af elementer i matricen, hvilket dermed vil sige antallet af samples. Koden for at gøre dette kan ses i listing 2

Listing 2: Print af antal samples

```
1 length(y(s1).sample) % = 4213759
2 length(y(s2left).sample) % = 8753617
3 length(y(s2right).sample) % = 8753617
4 length(y(s3).sample) % = 1270957
```

Her fra kommer jeg frem til at s1 indeholder 4.213.759 samples, s2 (left og rigth) indeholder 8.753.617 samples og s3 indeholder 1.270.957 samples. Dette lade til at være rigtigt, da jeg kan se s2 f.eks. varer meget længere end s3.

3 Opgave 2

For at plotte alle 3 signaler med rigtige labels og titler, bruges funktionerne beskrevet i 3. For at gøre MATLAB koden mere overskuelig, er der brugt et for-loop på min struct, så jeg kan loope igennem hvert signal, og plottet dem med de allerede forudbestemte varaibler. Alle plots tegnes på figur f1, for overskuelighed.

Listing 3: Plot af s1, s2 og s3

```
1  %% Exercise 2
2  f1 = figure;
3
4  for i = 1:length(y)
5      subplot(2,2,i)
6      plot(y(i).time, y(i).sample)
7      title(y(i).name)
8      xlabel("Time(s)")
9      ylabel("Amplitude(~)")
10 end
```

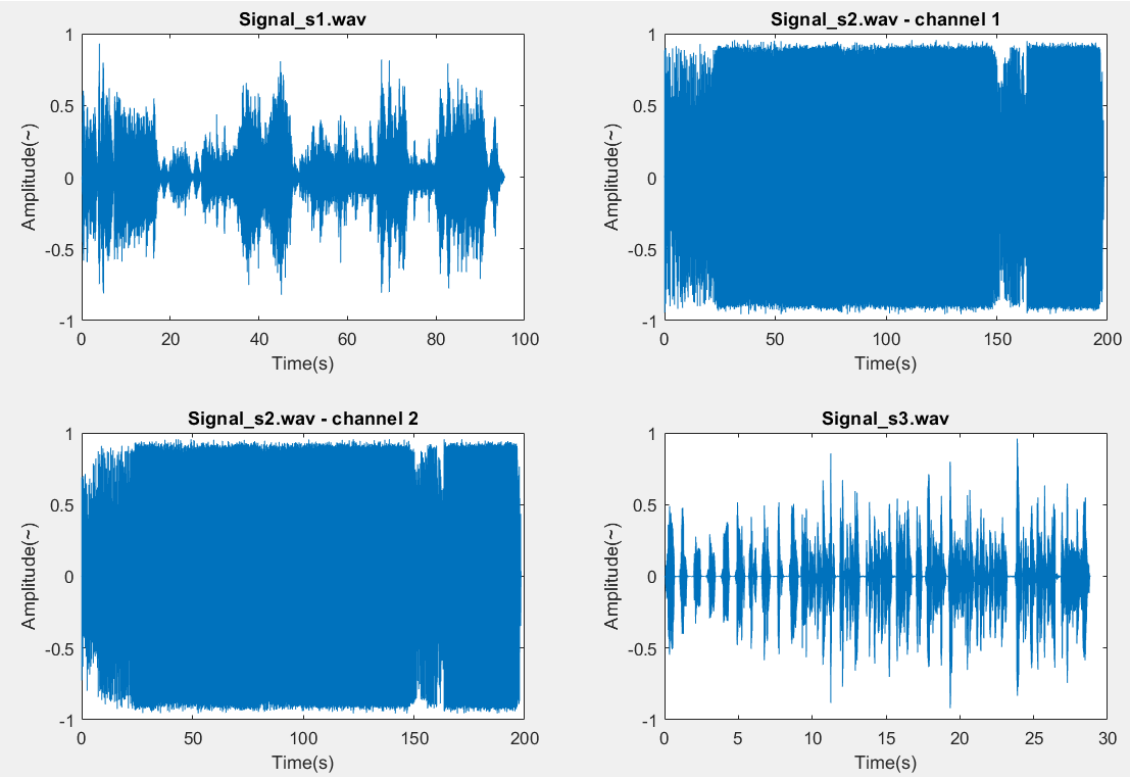

Figure 1: Plot af alle 3 signaler

4 Opgave 3

For at dfinde max, min, gennemsnit (mean), rms og effekt for alle 3 signaler, kan jeg kalde de forudbestemte værdier, jeg beregnede i listing 2. Et udsnit af hvordan de blev beregnet kan ses på listing 4.

Listing 4: Udsnit af beregning af max, min, gennemsnit (mean), rms og effekt

```
1 for i = 1:length(y)
2     y(i).max = max(y(i).sample);
3     y(i).min = min(y(i).sample);
4     y(i).mean = mean(y(i).sample);
5     y(i).rms = rms(y(i).sample);
6     y(i).effect = sum(y(i).sample.^2);
7 end
```

Table 1: Værdier for opgave 3

signal	max	min	mean	rms	effekt
s1	0.930	-0.820	≈ 0	0.119	60557.9
s2left	0.955	-0.955	≈ 0	0.338	1001787.6
s2right	0.955	-0.955	≈ 0	0.354	1102211.3
s3	0.960	-0.919	≈ 0	0.091	10738.9

5 Opgave 4

Ligesom i opgave 3, er crest allerede beregnet i listing 2. Til at beregne crest-faktoren, bruges formlen som beskrevet i ligning 1. Da der tage $20 \cdot log_{10}$, kan det antages at crest-faktoren vil være i decibel. Det på listing 5 kan det også ses, hvordan crest-faktoren er beregnet i MATLAB.

$$20 \cdot log_{10} \frac{x_{peak}}{x_{rms}} \tag{1}$$

Listing 5: Bergning af crest-faktor for alle 3 signaler

```
1 for i = 1:length(y)
2    y(i).crest = 20*log10(y(i).max/y(i).rms);
3 end
```

Table 2: Værdier af crest-faktor

signal	crest
s1	17.80 dB
s2left	9.02 dB
s2right	8.60 dB
s3	20.37 dB

I tabel 2 ses beregningen for de 3 signalers crest-faktor. Her ses der, at både s1 og s2 har en cirka dobbelt så stor crest-faktor, som s2 (både right og left). Svaret på hvorfor, kan findes i udregningen for crest-faktoren, da foruden omregningen til dB, tages x_{peak}/x_{rms}. Alle signalernes peak ligger omkring 0.95. Den store forskel er deres rms-værdi, hvor s2 har om rms-værdi på cirka 0.33, mens s1 og s3 begger ligger på en rms-værdi lige omkring 0.1. Herved har vi at x_{peak} divideres med et stører tal for s2, end for s1 og s3. At s2 har en lavere crest-faktor giver meget god mening, når crest-faktoren beskriver hvor ekstrem forskellen er mellem peaks og rms af signalet. Da s2 er elektronisk musik, med masser af "gang i den", må vi dermed også kunne forvente en høj rms. Omvendt er der ikke lige så meget gang i den, i et signal med Mozart musik, eller tale.

6 Opgave 5 og 6

Jeg vil nu forsøge at nedsample mit siganl s1 med en faktor på 4. Dette gøres med funktionen decimate, der ganske simpel fjerne hvert 4 sample, fra siganlet s1. Her er jeg selvfølgelig opmærksom på, der derfor også er en ny afstand mellem hvert sample, så jeg bliver nødt til at beregne tiden igen, med 1/4 sample rate. Dette kan ses her under på listing 6

Listing 6: Nedsampel af s1

```
s1reS = 5;
y(s1reS).sample = decimate(y(s1).sample, 4);
y(s1reS).time = [0:y(s1reS).nS-1]*(1/(Fs/4));

%% exercise 5
figure
subplot(1,2,1)
plot(y(s1).time(1:200), y(s1).sample(1:200))
title(y(s1).name)
xlabel("Time(s)")
ylabel("Amplitude(~)")
subplot(1,2,2)
plot(y(s1reS).time(1:200/4), y(s1reS).sample(1:200/4))
title(y(s1reS).name)
xlabel("Time(s)")
ylabel("Amplitude(~)")

% Listen to exercise 5
%soundsc(y(s1).sample, Fs);
%soundsc(y(s1reS).sample, Fs/4);
```

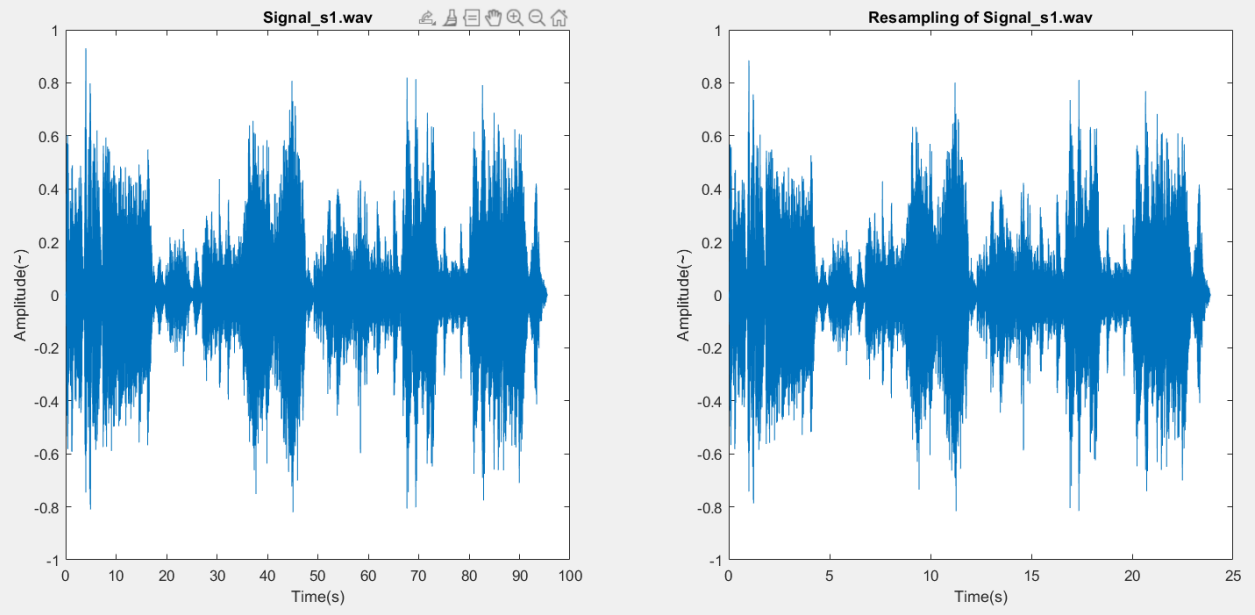

Figure 2: Resampling af signal s1

På plottet i figur 2 ses det ikke særlig tydeligt hvad, forskellen egentlig er belvet på de 2 signaler. Der er derfor taget et billede, hvor antallet af samples er taget meget længere ned, som det kan ses på figur 3. Her ses det mere tydeligt hvad decimate() har gjort. Da hvert fjerde sample er væk, er der tydeligt mistet en masse detaljer i signalet. F.eks. kan det ses at spændet af amplituder nu er -3.5 - 1.5, hvor der før var -6 - 2.

Lyttes der til signalet, hører man forskellen tydeligt. Det nye signal lyder meget komprimeret, som om det er en dårlig .mp3 fil, der afspilles på en Nokia 3210.

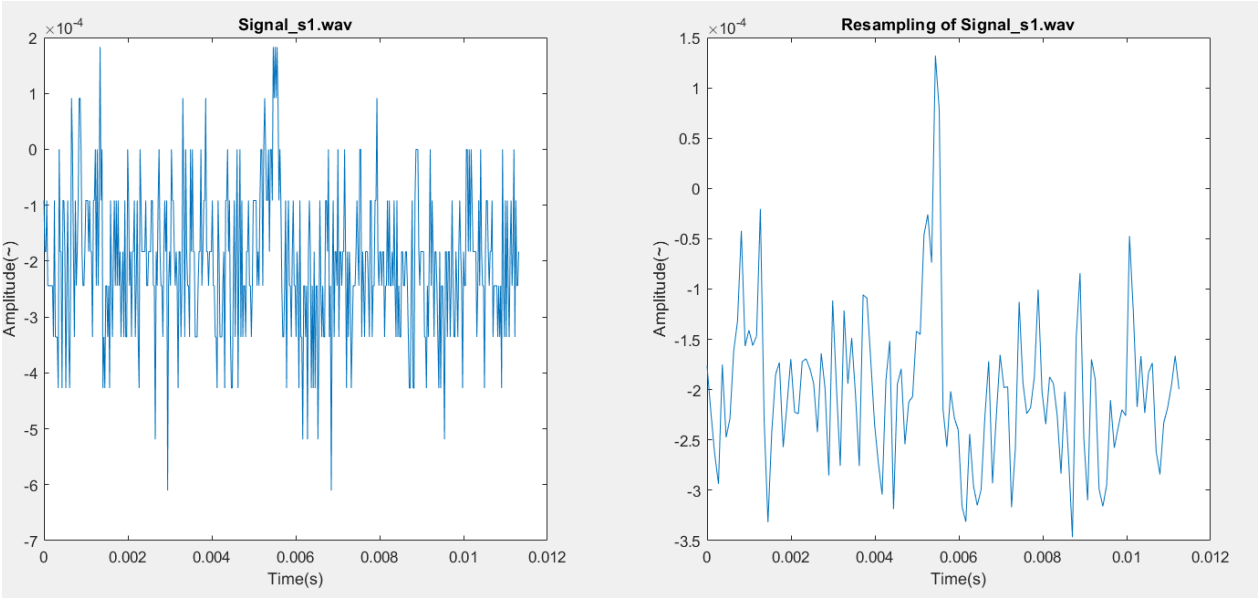

Figure 3: Zoom af resampling

7 Opgave 7

Til at sammenligne de channels på s2, er der først plottet begge channels ved siden af hindanden på figur 4. Da dette ikke har givet nogen stor information, er der derfor lavet et plot af en korter tidsperiod, et tilfædligt sted i signalet. Her ses en lille forskel, hvor amplituden er lidt større generelt på channel 2, men ikke noget tydeligt. Lyttes der til de 2 signaler, kan det være svært at spotte forskellen, da de egentlig er meget ens. Eneste forskel jeg har været i stand til at finde er, at der på channel 1 (left) kan findes guitar riffet, mens der på channel 2 (right) kan findes baselinen.

Listing 7: Kode for plot af s2left og s2 right

```
figure
subplot(1,2,1)
plot(y(s2left).time, y(s2left).sample)
title(y(s2left).name)
xlabel("Time(s)")
ylabel("Amplitude(~)")
subplot(1,2,2)
plot(y(s2right).time, y(s2right).sample)
title(y(s2right).name)
xlabel("Time(s)")
ylabel("Amplitude(~)")
```

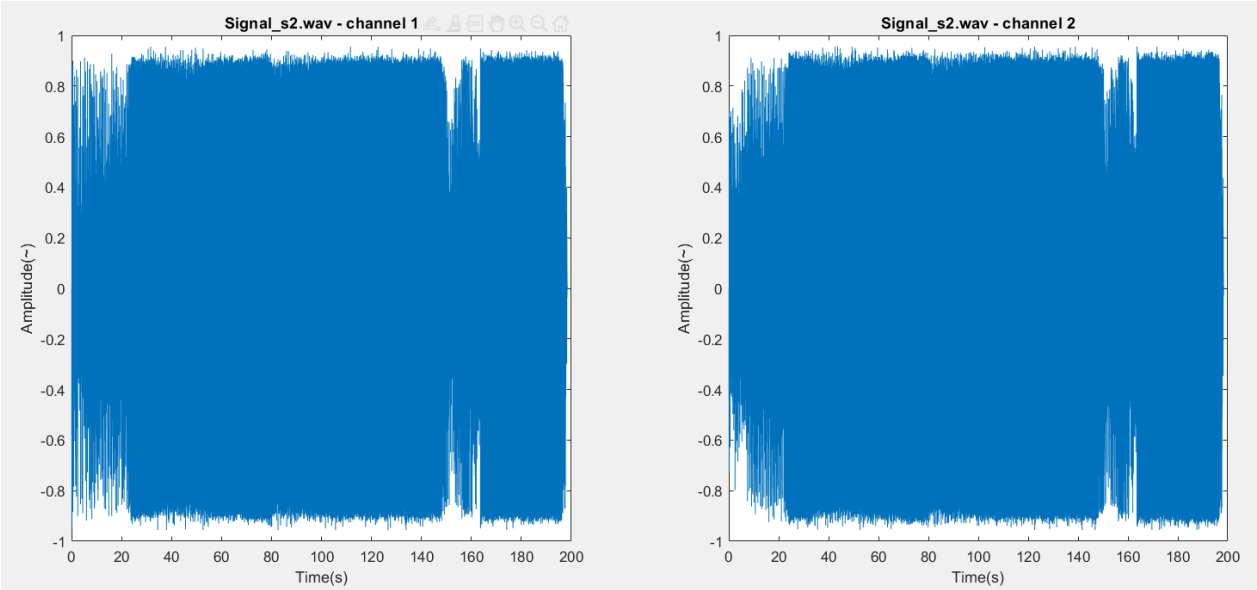

Figure 4: Plot af begge channels for s2

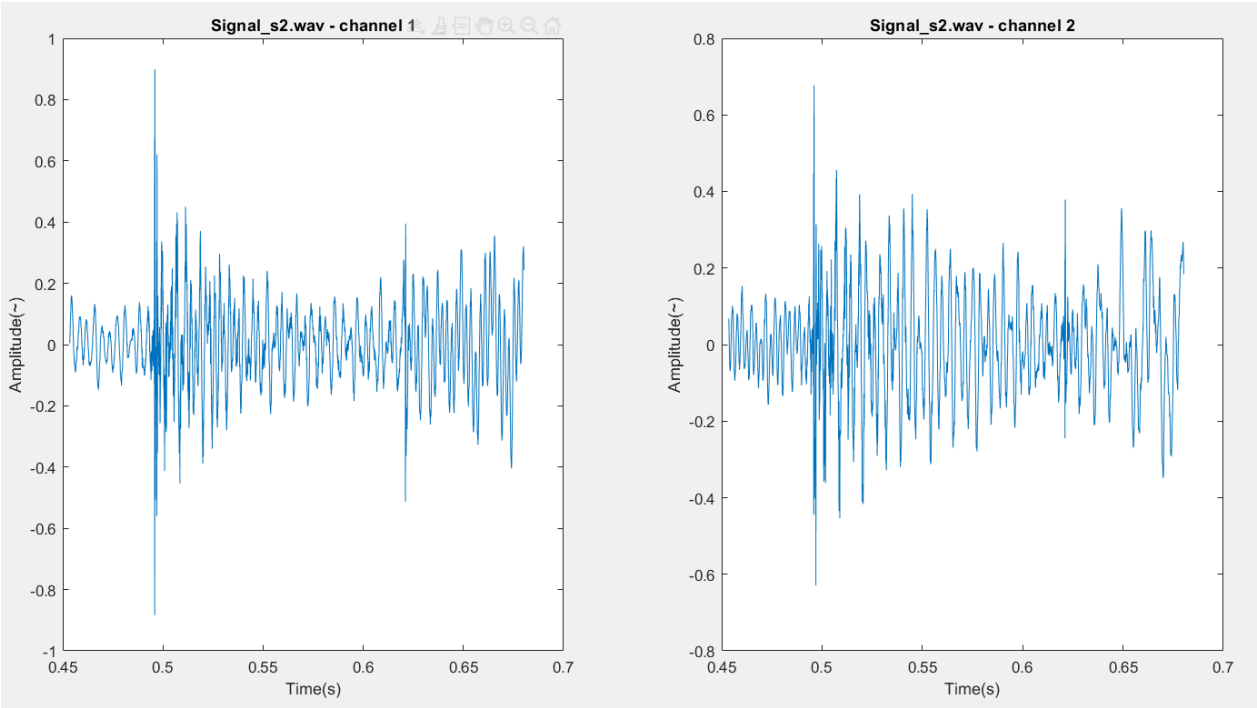

Figure 5: Plot af begge channels for s2, Zoom

8 Opgave 8

Til at kvantifiseret channel 1 (left), er der brugt den givne funktion quantaziseN() fra Blackboard. Når der kvantifiseres, fjernes detaljerne mellem amplituderne, så hvert sample af amplituden, kan repræsenteres med et færer antal bits. Her er valgt at kvantifisere s2 med 4 bits. Koden til dette kan ses i listing **??**

Listing 8: Kvantifisering af s2label

```
%% Exercise 8
%Using BB function to quantize to 4 bits
s2quan = 6;
y(s2quan).sample = quantizeN(y(s2left).sample, 4);
soundsc(y(s2right).sample, Fs) % uncomment for sound
figure
subplot(1,2,1)
plot(y(s2left).time(200000:205000), y(s2quan).sample(200000:205000))
xlabel("Time(s)")
ylabel("Amplitude(~)")
title("s2 - Left channel");
subplot(1,2,2)
plot(y(s2left).time(200000:205000), y(s2left).sample(200000:205000))
xlabel("Time(s)")
ylabel("Amplitude(~)")
title("s2 kvantifiseret 4bit - Left Channel");
```

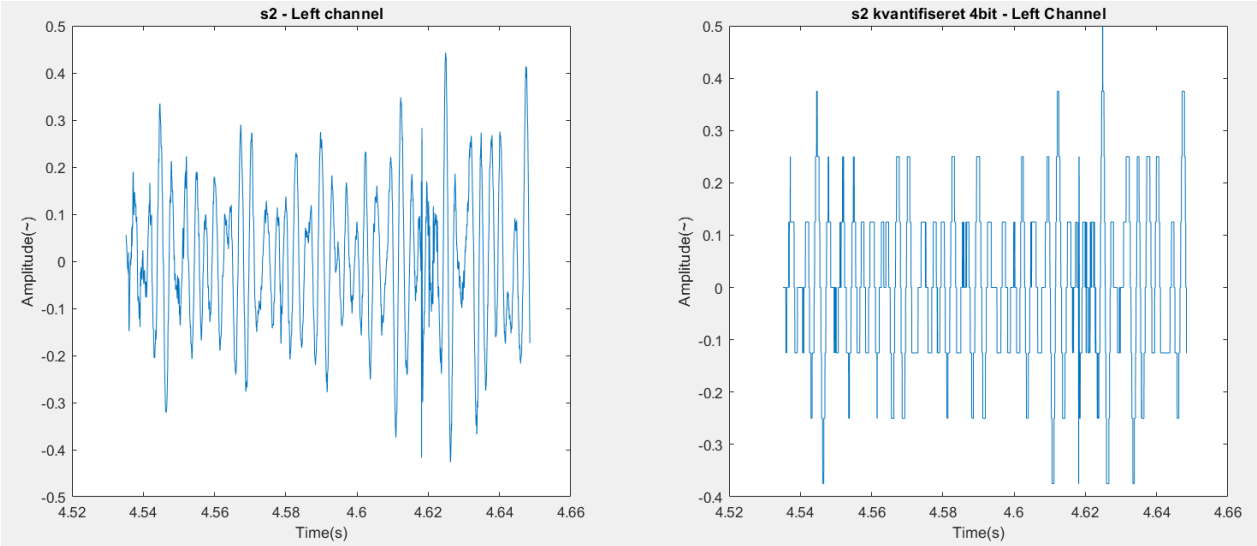

Figure 6: s2 kvantifiseret

På figur 6, som er et lille udsnit af både s2 og s2 kvantifiseret, ses tydeligt hvad quantaziseN har gjort ved signalet. Hvor amplituden i s2 left normalt repræsenteres med et så stort antal bits, det kanp er muligt at se steps imellem, kan der på det nye signal tydeligt ses de individuelle steps. Dette skyldes der samples med 4 bit, hvilket kun giver 16 mulige værdier for amplituden. Lytter man til signalt, kan der også høres hvordan

kvaliteten af musikken er faldet tydeligt. Mest tydeligt er det for instrumenterne, der har mistet en stor del af deltajerne, og nærmest cutter ud jævnligt. Modsat er sangerens stemme stadig forholdsvist tydeligt. Musikken lyder meget som om det er ventemusik, der afspilles når man venter på at komme igennem til f.eks. telefon kundeservice. Dette kunne være et hint til hvordan telefonlyd komprimeres.

9 Opgave 9

For at få en eksponentielt aftagning på s3, kræver det at s3 ganges med en "envelope" der kan fungere som der ønskes. Her at der på den sidste 1/3 af s3, skal aftages eksponentielt, ned til en amplitude på 5% af den originale amplitude.

$$y(x) = b \cdot a^x x = 2 \tag{2}$$

Listing 9: Fade out af s3label

```
%% Exercise 9
% Tage sidste tredjedel af s3
t = y(s3).time(1:y(s3).nS/3+1);
% Eksponentielt aftagende: b*a^x
% b = Skæring med y i (0,1)
% a = (y2/y1)^1/(x2-x1)
a = (0.05/1)^(1/(t(end)-t(1)));
% Matrice der er eksponentielt aftagenende
% Starter i 1 og går til 0.05 ved 1/3 af time
e = 1*a.^t;

% s3fade
s3fade = 7;
y(s3fade).sample = y(s3).sample;
y(s3fade).time = y(s3).time;
% Påfør e på sidste 1/3 af samples
y(s3fade).sample(length(y(s3fade).sample)*2/3:end)
   = e.*y(s3fade).sample(length(y(s3fade).sample)*2/3:end);

figure
plot(y(s3fade).time, y(s3fade).sample, Fs)
%soundsc(y(s3fade).sample)
```

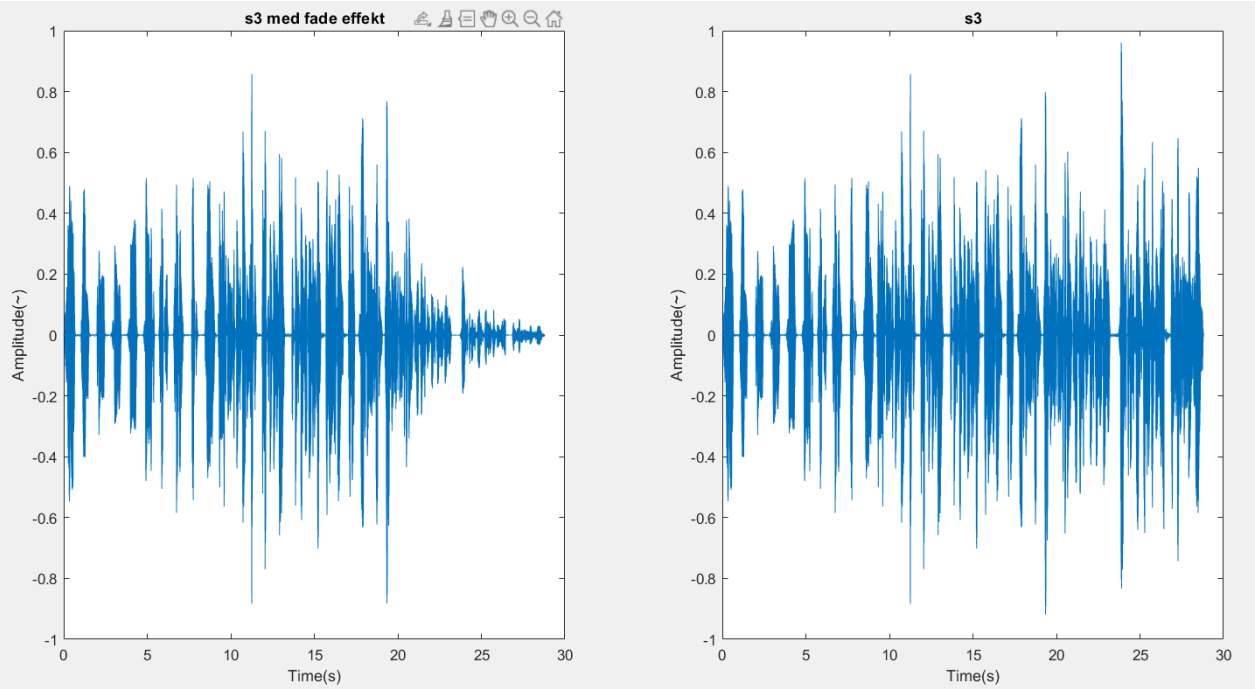

Figure 7: s3 med fade effekt

10 Opgave 1.15