Miniprojekt 1 - Sampling og Analyse af Digitale Audiosignaler

Digital Signalbehandling

Gruppe 4

Navn	Studienummer
Oskari Tuormaa	201808678
Thomas Kloster Ulriksen	201806733
Morten Haahr Kristensen	201807664

Ingeniørhøjskolen Aarhus Universitet

18. September 2019

Indhold

Indledning

I denne journal præsenteres gruppe 4's resultater ved gennemførelse af case-projekt 1 i kurset Digital Signalbehandling. Case-projekt 1 fokuserer på kvantisering og sampling af digitale lydsignaler, og tager udgangspunkt i en udleveret lydfil. Resultaterne fremført ved gennemarbejdning af projektet diskuteres kort og summeres til sidst i en samlet konklusion. Alle kodesekvenser benævnes som listings gennem journalen, og den samlede MATLAB kode ses i appendikset.
I tabel 1 ses hvilke opgaver de enkelte medlemmer har bidraget til. Det skal dog nævnes, at gruppemedlem William er gået på orlov under forløbet, og derfor ikke længere er tilmeldt kurset.

Navn	**Beskrivelse**
Morten Haahr Kristensen	Opsætning af Overleaf dokument samt mulighed for MATLAB kode deri. Korrigeret mangler i kode. Skrevet tekst til: Indledning, opgave 1, 2 og 3.
Oskari Tourmaa	Første udkast til mange af opgaverne. Skrevet meget af koden sammen med Thomas. Skrevet tekst til: Opgave 4, 5, 6 og 7.
Thomas Kloster Ulriksen	Sørget for billeder til opgaverne. Skrevet meget af koden sammen med Oskari. Skrevet tekst til: opgave 8, 9, 10 og konklusionen.

Tabel 1: Ansvarsfordeling for DSB miniprojekt 1

Opgave 1

Datafilen "Opgave1_audiofil.mat"blev indlæst i projektet i MATLAB. Antal samples i audio-filen blev fundet, ved at skrive `length(audio)`. Hele koden til opgave 1 ses nedenfor:

```
1 load('Opgave1_audiofil.mat');
2 length(audio)
3 % ans = 1409024
4 n = 1:length(audio);
5 fs = 44100;
```

Listing 1: Samlet kode til opgave 1

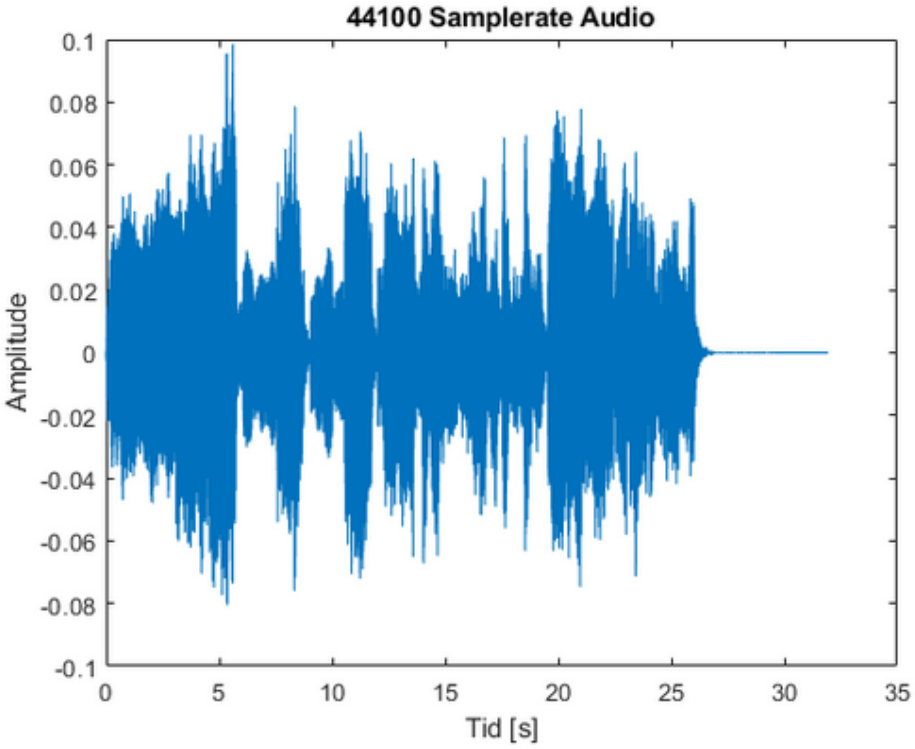
Som det kan ses i listing 1 linje 3, blev ans lig med 1.409.024, hvilket indikerer antallet af samplings. Der er desuden blevet erklæret en vektor n med samme antal pladser, som bruges til x-akse i kommende opgaver. Til sidst er samplingfrekvensen fs blevet erklæret.

Opgave 2

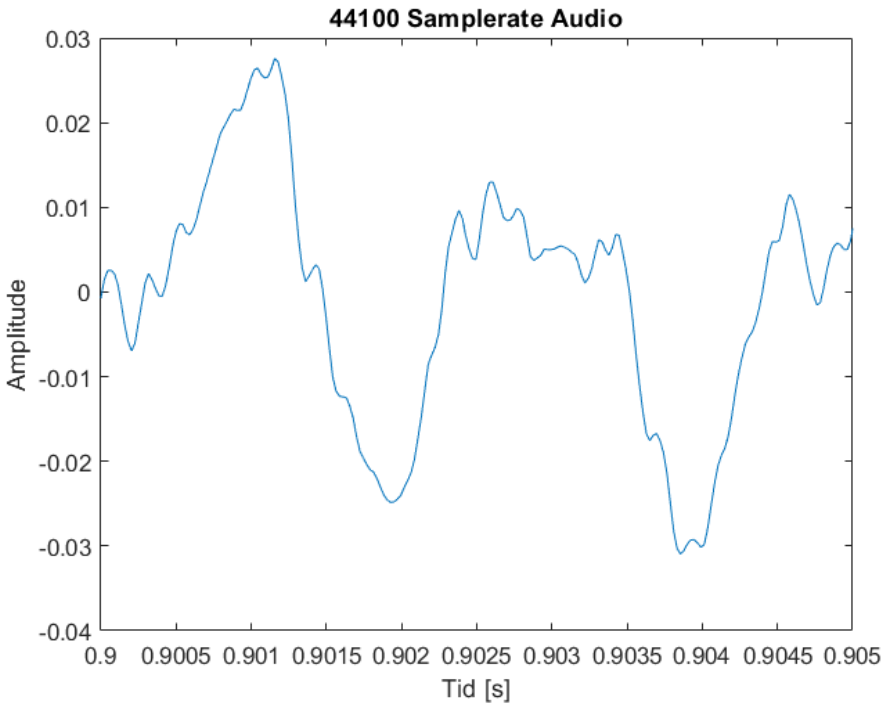
Et plot af det givne signal ses på figur 2. Funktionerne `xlabel()`, `ylabel()`, `title()` er brugt til at definere grafens akser og titel. Koden for at plotte og indstille akser ses i listing 2:

```
figure(1)
plot(n / fs, audio);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Audio');
xlim([0.9, 0.905]);
```

Listing 2: Samlet kode til opgave 2



Figur 1: Et plot af den givne audiofil



Figur 2: Et plot af den givne audiofil - her vist et selvvalgt interval fra 0.9s til 0.905s

Opgave 3

Til at finde audio signalets minimum, maksimum, gennemsnit, energi, RMS og effekt, bruges de indbyggede funktioner `max()`, `min()`, `mean()` og `rms()`. Hele koden til opgave 3 ses nedenfor:

```
1  maksimum = max(audio)
2  % maksimum = 0.0985
3
4  minimum = min(audio)
5  % minimum = -0.0804
6
7  gennemsnit = mean(audio)
8  % gennemsnit = -1.4752e-05
9
10 RMS = rms(audio)
11 % RMS = 0.0154
12
13 effekt = RMS^2
14 % effekt = 2.3731e-04
15
16 energi = effekt * length(audio)
17 % energi = 334.3711
```

Listing 3: Samlet kode til opgave 3

Det ses i listing 3 at effekt og energi ikke findes ved hjælp af funktioner. Disse kan udledes af RMS-værdien,

hvilket kan ses når formlerne undersøges nærmere.

$$RMS = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(x(n))^2}$$
$$P = \frac{1}{N}\sum_{n=0}^{N-1}(x(n))^2$$
$$E = \sum_{n=0}^{N-1}(x(n))^2$$

Hvor P er effekt, E er energi, N er antallet af samples og x er funktionen der regnes på. Det ses ud fra funktionerne at effekt og energi kan findes således:

$$P = RMS^2$$
$$E = P \cdot N$$

Vi vælger at finde effekt og energi på denne måde, da der derved kun skal laves udregninger over hele audio signalet èn gang. På denne måde sparer vi beregningstid.

Opgave 4

For at finde signalets crest factor findes først peak-værdien. Crest factoren beskriver nemlig signalets forhold mellem dets maksimale peaks og RMS'en. Peak-værdien beregnes da først som den numeriske værdi af audiosignalet og benytter Matlab-kommandoen, max, til at beregne peak-værdien:

```
1 peak = max(abs(audio))
2 %ans = 0.0985
3 crestfactor = peak / RMS
4 %ans = 6.3968
```

Listing 4: Samlet kode til opgave 4

Som det ses af listing 4, må crest factoren anses som værende forholdsvis stor, da den maksimale peak er over 6 gange større end RMS'en. Dette giver dog også mening, da der i musik som oftest er mange pauser mellem tonerne hvor signalniveauet er meget lavt.

Opgave 5

Audiosignalet skal nedsamples med en faktor 4, svarende til at samplefrekvensen bliver 4 gange lavere. Den nye sampling frekvens bliver:

```
1 fs_nedsamplet = fs / 4
2 %ans = 11025
```

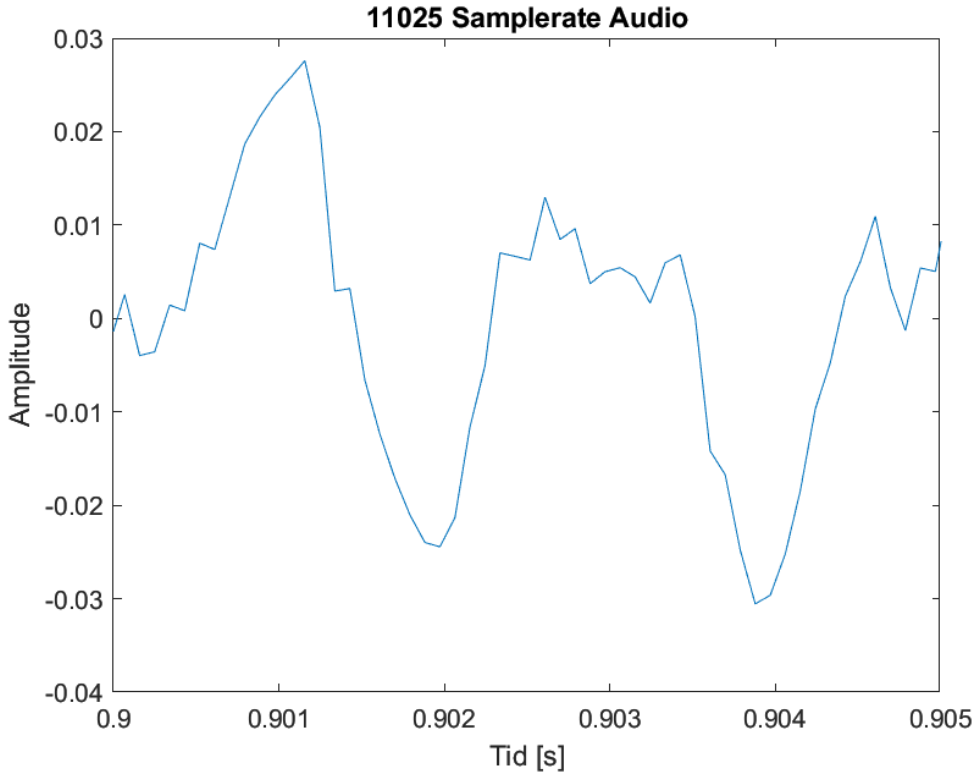
Listing 5: Beregning af den nedsamplede frekvens

Den nye nedsamplede frekvens benyttes til at plotte audiosignalet igen. Her må forventes at det nedsamplede signal bliver grovere og med færre nuancerede peaks:

```
plot(n([1:4:length(n)]) / fs, audio([1:4:length(audio)]));
xlabel('Tid [s]');
ylabel('Amplitude');
title('11025 Samplerate Audio');
xlim([0.9, 0.905]);
```

Listing 6: Kode til plot af nedsamplet audiosignal

Ved compiling af koden på listing 6, fås følgende plot af det nedsamplede audiosignal:



Figur 3: Et plot af den nedsamplede audiofil - her vist et selvvalgt interval fra 0.9s til 0.905s

Som det fremgår af figur 3, blev det nedsamplede signal som forventet. En del af de frekvenser forekommer mellem samplingstidspunkterne, hvilket gør signalet grovere og mindre nuanceret.

Opgave 6

Ved sammenligning af den nedsamplede audiofil med den orignale er det tydeligt at høre, hvordan detajler i lydbilledet går tabt. Det lyder som om signalet går gennem et filter.

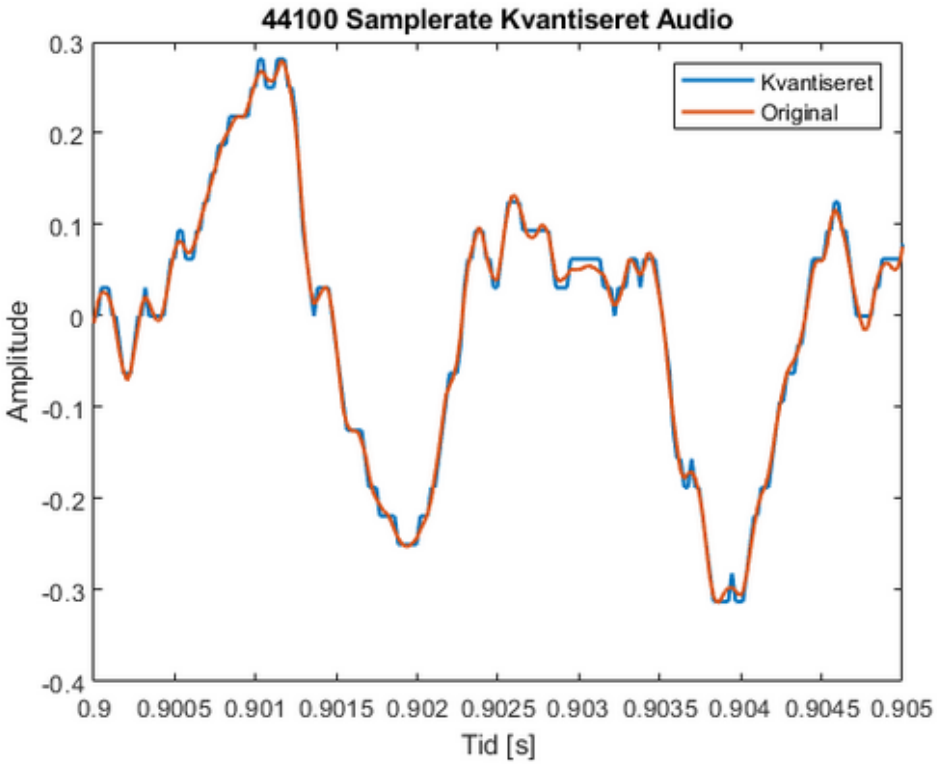
Opgave 7

I denne opgave skal audiosignalet kvantiseres med 4 bit og plottes. Vi har fået givet funktionen `quantise_n_bits` som bruges til at kvantisere. I listing 7 ses koden for både kvantisering og efterfølgende et plot af audiosignalet. Plottet af audiosignalet ses på figur 4.

```
audio_quantised = quantise_n_bits(audio * (1/maksimum), 4);
audio_quantised = audio_quantised;
figure(3);
plot(n / fs, audio_quantised, n / fs, audio, 'LineWidth', 1.5);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Kvantiseret Audio');
legend('Kvantiseret', 'Original')
xlim([0.9, 0.905]);
```

Listing 7: Samlet kode til opgave 7

Som det kan ses på figur 4 bliver det kvantiserede signal betydelig mere kantet end det originale. Vi kan derfor forvente en mere distorted lyd med forstærket støj.



Figur 4: Et plot af både det kvantiserede og orignale normaliserede audiosignal - her vist et selvvalgt interval fra 0.9s til 0.905s

Opgave 8

Ved sammenligning af det kvantiserede audiosignal med det originale audiosignal, kan det tydeligt høres at støjniveauet er blevet løftet mærkant. Dette giver god mening da hvis støjen står og skifter mellem to kvantiserings niveauer, vil den blive forstærket.

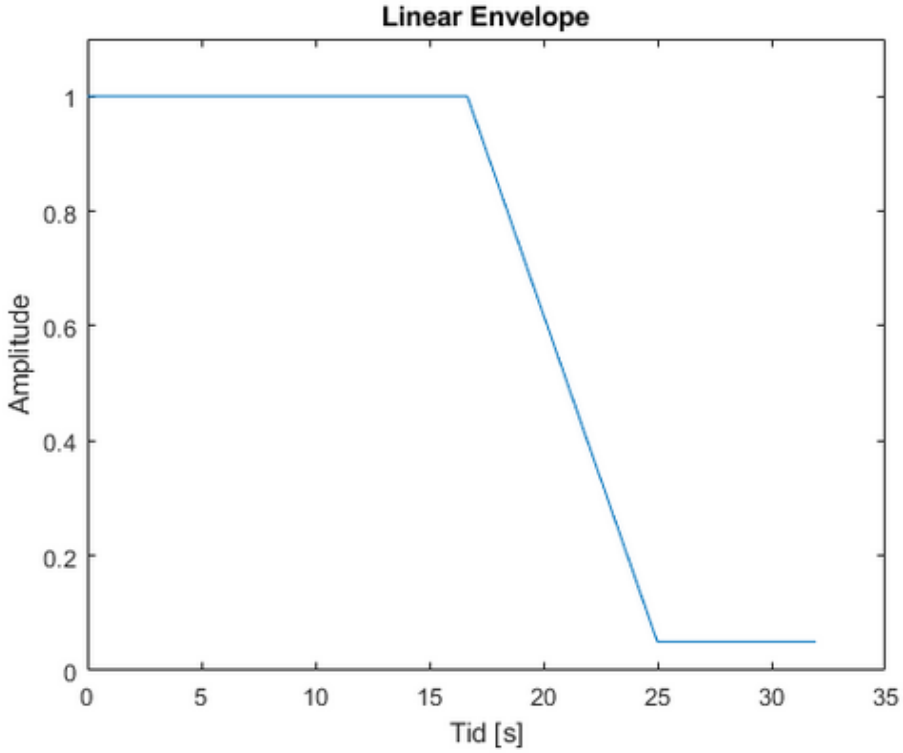
Opgave 9

En envelope skal tilføjes til audiosignalet over den sidste tredjedel af filen. Fremgangsmåden her bliver at en vektor ganges på audiosignalet. I den første 2/3 af vektoren indsættes 1'taller da der ikke skal ændres på signalet. Den sidste tredjedel af vektoren indsættes en tal række fra 1 til 0.05 da audiosignalet skal ende med 5% volumen. Koden for processen ses på listing 8.

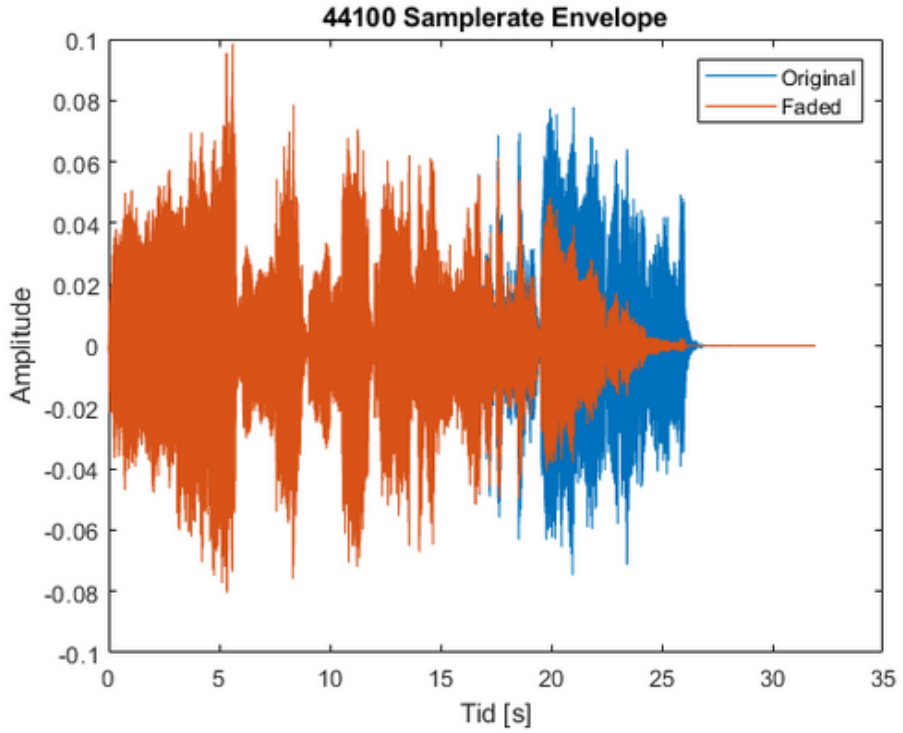
```
% Amount of samples till audio goes silent
n_samp = 25*fs;

% Fill the vector envel with 1 with a size of n_samp*(2/3)
envel = zeros(1, floor(n_samp * (2/3))) + 1;

% dA is the amplitude change per step to linearly go from 1 to 0.05 over
% n_samp*(1/3) steps
dA = 0.95 / (floor(n_samp * (1/3)));
envel = [envel, 1:-dA:0.05];

% Fill the rest of envel with 0.05
envel = [envel, zeros(1, length(audio) - length(envel)) + 0.05];

% Multiply envel elementwise with the audio to fade the audio out
audio_envelope = envel.*audio';


figure(4);
plot(n / fs, envel);
xlabel('Tid [s]');
ylabel('Amplitude');
title('Linear Envelope');
ylim([0,1.1]);

figure(3);
plot(n / fs, audio, n / fs, audio_envelope);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Envelope');
legend('Original', 'Faded');
soundsc(audio_envelope, fs);
```

Listing 8: Samlet kode til opgave 9

Først initialiseres variablen **envel** som en vektor med længden der svarer til 2/3 af 25 ms. Vi tager 2/3 af 25 ms i stedet for hele signalet, da audio alligevel slutter omkring 25 ms. Derefter tilføjes den sidste 1/3 som et linært fald fra 1 til 0.05. Da **envel** nu kun fylder 25 ms af signal, fyldes den sidste del af signalet op med 0.05. Selve envelopen kan ses for sig selv på figur 5. Envelopens indflydelse på audio signalet kan ses på figur 6.



Figur 5: Grafen viser det envelope vi ganger på audiosignalet



Figur 6: Grafen viser en sammenligning mellem envelope-signal og det originale audiosignal

Som det ses på figur 6 fades audio signalet ud over den sidste 1/3 af signalets længde.

Konklusion

Vi er blevet klogere på den effekt samplerate og kvantisering kan have på et audio signal. Gennem opgaven har vi nedsamplet et givet signal og studeret de effekter det har på især lydkvaliteten af signalet. Det nedsamplede signal er blevet kvantiseret med et forholdsvist lavt antal bit, og støjeffekterne af denne er blevet undersøgt. Alt i alt kan man konkludere, at nedsampling og kvantisering har stor betydning for signalets kvalitet, og i vores specifikke tilfælde kompromiteredes de højfrekvente svingninger og signal-to-noise ratioen for signalet.

Appendix

```
%% Opgave 1
clear all
load('Opgave1_audiofil.mat');
length(audio)
n = 1:length(audio);

fs = 44100;

%% Opgave 2
figure(1)
plot(n / fs, audio);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Audio');
xlim([0.9, 0.905]);

figure(5)
plot(n / fs, audio);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Audio');

%% Opgave 3
maksimum = max(audio)
minimum = min(audio)
gennemsnit = mean(audio)
RMS = rms(audio)
effekt = RMS^2
energi = effekt * length(audio)

%% Opgave 4
peak = max(abs(audio))
crestfactor = peak / RMS

%% Opgave 5
figure(2)
fs_nedsamplet = fs / 4
plot(n([1:4:length(n)]) / fs, audio([1:4:length(audio)]));
xlabel('Tid [s]');
ylabel('Amplitude');
title('11025 Samplerate Audio');
xlim([0.9, 0.905]);

%% Opgave 6
soundsc(audio, fs);
pause(27);

soundsc(audio([1:4:length(audio)]), fs_nedsamplet)
pause(27);

%% Opgave 7
audio_quantised = quantise_n_bits(audio / maksimum, 5);
audio_quantised = audio_quantised;
figure(3);
plot(n / fs, audio_quantised, n / fs, audio / maksimum, 'LineWidth', 1.5);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Kvantiseret Audio');
legend('Kvantiseret', 'Original');
xlim([0.9, 0.905]);
```

```
%% Opgave 8
soundsc(audio_quantised, fs)
pause(27);

%% Opgave 9
% Amount of samples till audio goes silent
n_samp = 25*fs;

% Fill the vector envel with 1 with a size of n_samp*(2/3)
envel = zeros(1, floor(n_samp * (2/3))) + 1;

% dA is the amplitude change per step to linearly go from 1 to 0.05 over
% n_samp*(1/3) steps
dA = 0.95 / (floor(n_samp * (1/3)));
envel = [envel, 1:-dA:0.05];

% Fill the rest of envel with 0.05
envel = [envel, zeros(1, length(audio) - length(envel)) + 0.05];

% Multiply envel elementwise with the audio to fade the audio out
audio_envelope = envel.*audio';


figure(4);
plot(n / fs, envel);
xlabel('Tid [s]');
ylabel('Amplitude');
title('Linear Envelope');
ylim([0,1.1]);

figure(3);
plot(n / fs, audio, n / fs, audio_envelope);
xlabel('Tid [s]');
ylabel('Amplitude');
title('44100 Samplerate Envelope');
legend('Original', 'Faded');
soundsc(audio_envelope, fs);
```

Listing 9: Samlet kode til hele miniproject 1