E3DSB miniprojekt 1 - Tidsdomæneanalyse

Janus Bo Andersen [1]

15. september 2019

[1] ja67494@post.au.dk

Indhold

1. Indledning

Dette første miniprojekt i E3DSB behandler tre lydsignaler med analyser i tidsdomænet. Opgaven er løst individuelt. Dette dokument er genereret i Matlab med en XSL-template. Matlab-kode og template findes på `https://github.com/janusboandersen/E3DSB`. Følgende lydklip benyttes

Signal	Skæring	Genre	Samplingsfrekv.
s_1	Spit Out the Bone	Thrash-metal	44.1 kHz
s_2	The Wayfaring Stranger	Bluegrass	96 kHz
s_3	Svanesøen	Klassisk	44.1 kHz

Tabel 1.1: 3 signaler behandlet i analysen

2. Analyser

Før analyser ryddes der op i `Workspace`.

```
clc; clear all; close all;
```

2.1 Afspilning af lydklip

Filen med signaler åbnes med `load`. Signaler kan afspilles med `soundsc(signal, fs)`. Samplingsfrekvensen f_s sættes efter værdi i tabel 1.1. Samplingfrekvenser for de tre signaler er inkluderet i `.mat`-filen.

```
load('miniprojekt1_lydklip.mat');    % åbn .mat-fil
soundsc(s1, fs_s1);                  % playback startes sådan her
clear('sound');                      % stop playback
```

2.2 Bestemmelse af antal samples

Et sample er en værdi, eller sæt af værdier, fra et givent punkt i tid. Alle tre signaler er i stereo, så hver sample har to værdier.

Signalerne er repræsenteret som $N \times K$-matricer. Antallet af rækker, N, repræsenterer antallet af samples. N kan findes med `length(matrix)`. Antallet af søjler, K, er antallet af kanaler (værdier per sample). Samlet antal af værdier i matricen er NK, antaget at ingen er `NaN`.

N og K kan bestemmes på en gang via `[N, K] = size(matrix)`. Vi kan også bare benytte, at vi ved, at der er to kanaler, så $K = 2$.

Data samles i en tabel. Den kan udvides med signalernes afspilningstider.

Der er altså fx 1,323 millioner samples i signal s_1. Signal s_2, som dog har højere samplingsfrekvens, har 2,5 gange flere samples. De tre lydklip har afspilningstider på mellem 30 og 35 sek.

```
signaler = {'s1'; 's2'; 's3'};
N = [length(s1); length(s2); length(s3)];           % antal samples
K = [2; 2; 2];                                      % antal kanaler
M = N.*K;                                           % antal værdier
samplingsfrek = [fs_s1; fs_s2; fs_s3];              % f_s fra .mat-fil
tid = N./samplingsfrek;                             % spilletid i sek.
T = table(signaler, N, K, M, samplingsfrek, tid)    % vis en datatabel
```

```
T =

  3×6 table

    signaler        N         K        M         samplingsfrek    tid
    --------    ----------    -    ----------    -------------    ---

      's1'       1.323e+06    2     2.646e+06        44100        30
      's2'        3.36e+06    2      6.72e+06        96000        35
      's3'      1.4112e+06    2    2.8224e+06        44100        32
```

2.3 Plot af signal

Når vi skal plotte signalerne med en tidsakse i sekunder, bruges det at $t = nT_s = \frac{n}{f_s}$. Man bør plotte et diskret signal i et stem-diagram, dvs. `stem`-funktionen, men for at få noget mindre gnidret at se på, bruges `plot`. Til at danne akserne bruges Matlabs :-operator.

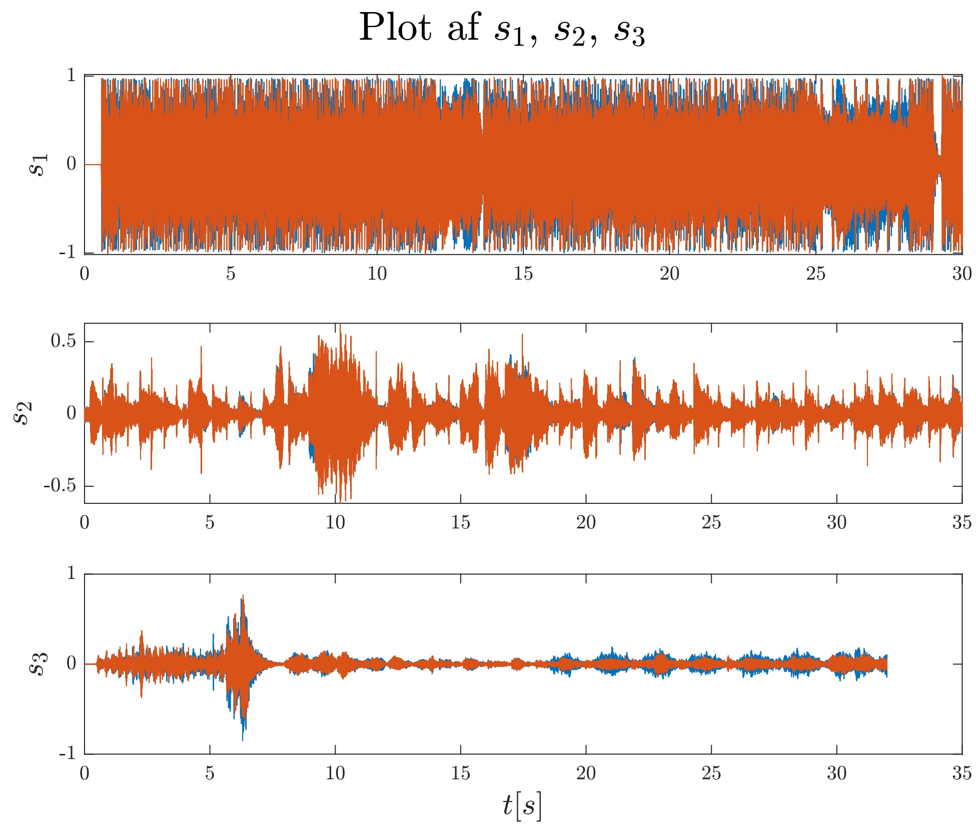
```
t1 = [0:1:N(1)-1]'/fs_s1;                     % søjlevektor, dog ej vigtigt
t2 = [0:1:N(2)-1]'/fs_s2;
t3 = [0:1:N(3)-1]'/fs_s3;

% der gøres lidt arbejde for at få et rent latex layout
set(groot, 'defaultAxesTickLabelInterpreter','Latex');
set(groot, 'defaultLegendInterpreter','Latex');
set(groot, 'defaultTextInterpreter','Latex');

figure(1)                                     % figur med 3 stablede subplots
```

```
subplot(3,1,1);
plot(t1,s1);                                    % signal 1
ylabel('$s_1$','Interpreter','Latex', 'FontSize', 15);
subplot(3,1,2);
plot(t2,s2);                                    % signal 2
ylabel('$s_2$','Interpreter','Latex', 'FontSize', 15);
subplot(3,1,3);
plot(t3,s3);                                    % signal 3
ylabel('$s_3$','Interpreter','Latex', 'FontSize', 15);
xlabel('$t [s]$','Interpreter','Latex', 'FontSize', 15);

% og en titel for hele diagrammet
sgtitle('Plot af $s_1$, $s_2$, $s_3$', 'Interpreter', 'Latex', 'FontSize', 20);
```



Plots viser ret tydeligt store forskelle i lydklippenes "intensitet". Forstået på den måde, at lydklippet med thrash-metal har en gennemgående høj amplitude (opleves som "højt"), i modsætning til fx det klassiske stykke. Nogle ville nok bare mene, at plottet over Metallicas nummer ligner "støj" :-).

Næste analyse kan måske give numeriske mål på disse visuelle observationer.

2.4 Min, max, energi og RMS

I dette afsnit beregnes forskellige mål på signalernes lydmæssige "karakter".

Overvejelser: Signalerne er i stereo (2 kanaler / søjler). Hvis vi har et system med to højttalere, giver det mening at betragte kanalerne separat (ikke sammenlagt). Altså, jeg analyserer kanalerne i forlængelse, som en mono serie med $M = 2N$ samples. Denne løsning bruges, fordi det er sådan et menneske med to ører og sæt hovedtelefoner ville opleve signalet :-). Det er også proportionalt til effekt og energiafsættelse i et system med to højttalere.

En sum eller et gennemsnit på tværs af kanalerne ville betyde, at kanaler ude af fase kunne cancellere/eliminere hinanden. Dette ville måske give mening som en simpel konvertering til mono, dvs. vi kunne beregne mål på hvad der ville ske i et simpelt mono-system.

Beregning: Minimum og maksimum findes med hhv. `min()` og `max()`. I tidsdomænet er effekten af et signal proportionalt til kvadratet på amplituden. For en sekvens $x(n) \in \mathbb{R}$, $n = 0, \dots, N-1$ defineres effekten som $x_{pwr}(n) = |x(n)|^2 = x(n)^2$. I diskret tid er energien i signalet summen af "effekterne", dvs. $E_x = \sum_{n=0}^{N-1} |x(n)|^2$. Dette er også det indre produkt $\langle x(n), x(n) \rangle$. RMS-værdien kan beregnes som kvadratroden af middeleffekten, dvs. $x_{RMS} = \sqrt{\frac{1}{N} E_x}$. Nu regnes alle serier så blot over $n = 0, \dots, 2(N-1)$ jf. overvejelserne ovenfor.

```
s1_vec = reshape(s1,[],1);       % Reshape matricer til søjlevektorer:
s2_vec = reshape(s2,[],1);       % De har nu hver M = 2N rækker og 1 søjle
s3_vec = reshape(s3,[],1);       % N, M er selvfølgelig forskellige for hver

minima = [min(s1_vec); min(s2_vec); min(s3_vec)];
maxima = [max(s1_vec); max(s2_vec); max(s3_vec)];
energi = [sum(s1_vec.^2); sum(s2_vec.^2); sum(s3_vec.^2)];      % kvadratsum
rms = [energi(1)/M(1); energi(2)/M(2); energi(3)/M(3)].^(1/2); % kv.rod

T = table(signaler, N, M, minima, maxima, energi, rms)          % resultater
```

```
T =

  3×7 table

    signaler        N             M           minima      maxima       energi         rms
    --------    ----------    ----------    --------    -------    ----------    --------

      's1'       1.323e+06     2.646e+06     -1.0166     1.0191    2.5336e+05     0.30944
      's2'        3.36e+06      6.72e+06    -0.61796    0.62791         34641    0.071797
      's3'      1.4112e+06    2.8224e+06    -0.85016    0.76907        5662.2     0.04479
```

Resultaterne (i tabellen) viser det, som plots også illustrerede: Der er mere energi i metal end i klassisk og bluegrass :-) Og højttalerne bliver varmere af at spille Metallica end af Tchaikovsky.

2.5 Venstre vs. højre kanal (for s_1)

Man kan eksperimentere lidt for at finde ud af hvilken kanal, der er højre, og hvilken der er venstre. Man kan jo fx fylde den ene kanal med nuller, og så se, hvad der "sker". Stereo bibeholdes ved at fastholde matricens $N \times K$-størrelse, men med en kanal "nullet".

```
s1_left_stereo = s1;
s1_left_stereo(:,2) = zeros(N(1),1); % Nuller "højre" via 2. kanal
soundsc(s1_left_stereo, fs_s1);      % Bingo, det virkede
clear sound;

s1_right_stereo = s1;
s1_right_stereo(:,1) = zeros(N(1),1); % Nuller "venstre" via 1. kanal
soundsc(s1_right_stereo, fs_s1);
clear sound;
```

Differensen mellem kanalerne kan også aflyttes. Vi tager venstre minus højre.

```
s1_diff_mono = s1(:,1) - s1(:,2);       % venstre minus højre
soundsc(s1_diff_mono, fs_s1);
clear sound;
```

Differensen mellem kanalerne giver en effekt af at lyden "kommer" et bestemt sted fra, rumligt/spatialt (eller evt. at der er en genklang). Fx vil en lille forsinkelse i den højre kanal snyde hjernen til at tro, at lyden kom fra et sted tættere på det venstre øre. Forsinkelse kan derfor benyttes til at "flytte" instrumenternes lyd i rummet.

I dette lydklip oplever jeg, at alle instrumenter er tilstede i både venstre og højre kanal, men i forskellig grad. Differensen afslører, at:

- Den hurtige lyd af J. Hetfields downpicking/strumming bevæger sig mellem kanalerne.
- Det gør lyden af L. Ulrichs lilletromme til dels også.
- Det giver en fornemmelse af at være omringet af lyden.
- Desuden er L. Ulrichs hi-hat placeret til venstre for midten på enkeltslagene, men til højre for midten på triple-slaget.

Hvis klippet havde været længere, havde vi også tydeligt hørt den fede og lidt mere melodiske del af guitarriffet (som starter ca. 40 sekunder inde) placeret i venstre kanal.

2.6 Nedsampling af signal (for s_1)

Der laves en nedsampling af signalet med en faktor 4. Funktionen `resample(signal, fs_ny, fs_gl)` benyttes.

```
fs_s1_ny = fs_s1 / 4;                          % reduktion med faktor 4
s1_downsample = resample(s1, fs_s1_ny, fs_s1);  % downsampling
txt = sprintf("Nyt antal samples: %d", length(s1_downsample));
disp(txt);

soundsc(s1_downsample, fs_s1_ny);              % afspil nyt klip
clear sound;
```

```
Nyt antal samples: 330750
```

Det høres tydeligt, at downsampling har reduceret lydkvaliteten. Klippet lyder nu mere som internetradio i 90'erne, eller en dårlig YouTube-video.

2.7 Fade-out med envelopes (for s_2)

Vi vil lave fade-out over den sidste tredjedel af signalet. Dvs. cirka de sidste 12 af de i alt 35 sekunder. Helt præcist skal indhyldningskurven påvirke de sidste 1,12 mio. samples. Altså $N_{env,2} = \frac{1}{3}N_2 = 3360000/3 = 1120000$. Der benyttes to forskellige metoder:

- Lineær envelope fra 100 til 5 pct.
- Eksponentielt aftagende envelope fra 100 til 5 pct.

Metoden bliver at lave envelopes med den ønskede længde, og så applicere dem på den sidste tredjedel af signalet.

2.7.1 Lineær envelope

Der skal over $N_{env,2}$ samples foretages en **lineær** "afskrivning". Funktionen er $f_{lin}(n) = -\alpha n$ for $n = 0, \ldots, N_{env,2} - 1$. Yderpunkterne sættes $f_{lin}(0) = 1$ og $f_{lin}(N_{env,2} - 1) = 0.05$. Det giver en hældning på $\alpha = -\frac{(0.05-1.00)}{N_{env,2}-1} = 8.48 \cdot 10^{-7}$.

Men det er naturligvis nemmere bare at bruge `linspace`...

```
N_env2 = N(2)/3;                           % antal samples der skal filtr.
lin_env2 = linspace(1.0, 0.05, N_env2)';   % lineær envelope
```

2.7.2 Eksponentiel envelope

Der skal over $N_{env,2}$ samples foretages en **eksponentiel** "afskrivning". Funktionen er $g_{exp}(n) = \exp(-\gamma n)$ for $n = 0, \ldots, N_{env,2} - 1$. Yderpunkterne sættes $g_{exp}(0) = 1$ og $g_{exp}(N_{env,2} - 1) = 0.05$. Det giver med lidt omskrivning en faktor på $\gamma = -\frac{\ln(0.05)}{N_{env,2}-1} = 2.67 \cdot 10^{-6}$.

```
gamma = -log(0.05)/(N_env2 - 1);           % nb. ln = log()
exp_env2 = exp(-gamma*[0:N_env2-1])';      % vektoriseret exp envelope
```

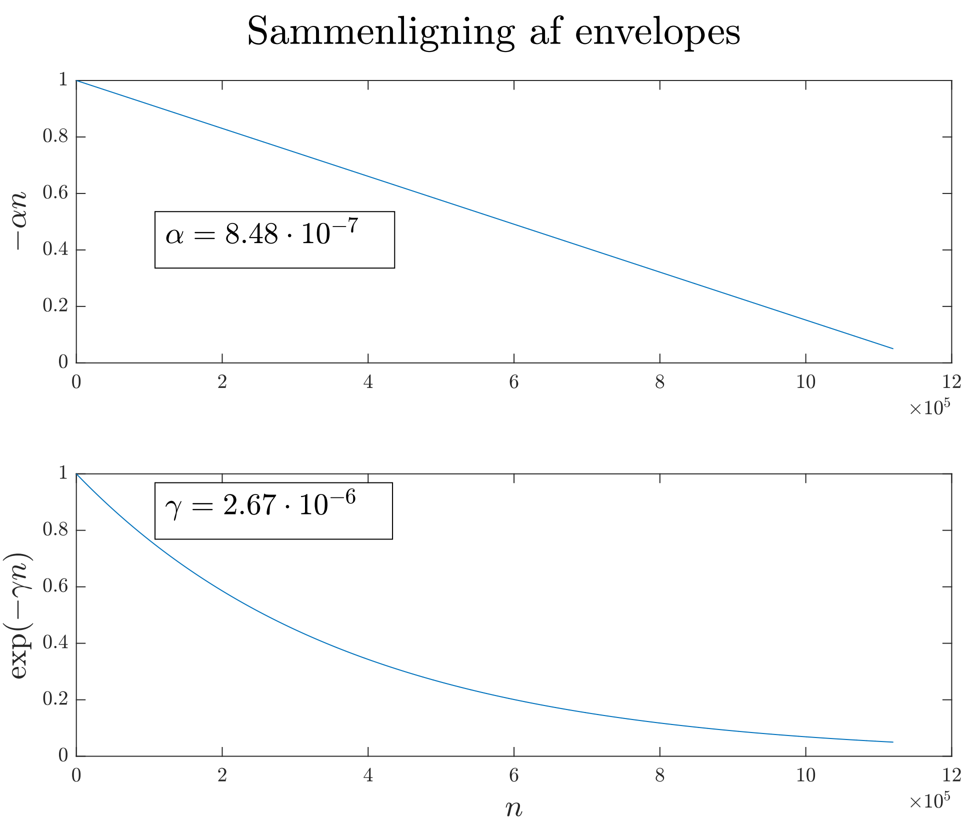
2.7.3 Sammenligning af envelopes

De to envelopes (indhyldningskurver) plottes, så vi kan se, om vi har fået hvad vi ønskede...

```
figure(2)
subplot(2,1,1);
plot(lin_env2);
ylabel('$-\alpha n$','Interpreter','Latex', 'FontSize', 15);
dim1 = [.2 .45 .3 .3];                        % Placering af annotation
str_lin = '$\alpha = 8.48\cdot10^{-7}$';
annotation('textbox',dim1,'Interpreter', 'Latex', 'String',str_lin,'FitBoxToText','on', 'FontSize', 15);

subplot(2,1,2);
plot(exp_env2);
ylabel('$\exp(-\gamma n)$','Interpreter','Latex', 'FontSize', 15);
str_exp = '$\gamma = 2.67\cdot10^{-6}$';
dim2 = [.2 .13 .0 .3];                        % Placering af annotation
annotation('textbox',dim2,'Interpreter', 'Latex', 'String',str_exp,'FitBoxToText','on', 'FontSize', 15);

xlabel('$n$','Interpreter','Latex', 'FontSize', 15);
sgtitle('Sammenligning af envelopes', 'Interpreter', 'Latex', 'FontSize', 20);
```



Envelopes påtrykkes signalet direkte, selvom man nok også kunne have brugt `filter`-funktionen.

```
pad_ones = ones([2*N_env2, 1]);          % pad med 1-taller når sig. ej ændr
tot_lin_fade = [pad_ones; lin_env2];     % sammensæt for hele serien
tot_exp_fade = [pad_ones; exp_env2];

% Fade påtrykkes hver kanal
s2_lin_fade = s2;
s2_exp_fade = s2;
for k=1:2
    s2_lin_fade(:,k) = s2_lin_fade(:,k) .* tot_lin_fade;  % påtryk lineær
    s2_exp_fade(:,k) = s2_exp_fade(:,k) .* tot_exp_fade;  % påtryk eksp.
end

% Afspil resultaterne
soundsc(s2_lin_fade, fs_s2);
clear sound;

soundsc(s2_exp_fade, fs_s2);
clear sound;
```

Det er nok smag og behag med de to forskellige typer. Jeg bryder mig bedst om den eksponentielle fade-out, fordi den hurtigere reducerer lydstyrken. Det mest naturlige ville nok være en logaritmisk fade, der matcher vores ørers og hjernes evne til at opfatte forskelle i lydniveauer, hvilket netop oftest er “efter” logaritimisk skala.

3. Konklusion

Dette miniprojekt har vist, hvordan man kan arbejde med digitale lydsignaler i Matlab.

Det er interessant, hvordan relativt simple matematiske metoder kan benyttes til at analysere og behandle digitale lydsignaler. Matlab gør arbejdet nemt for os.