Aktiv Støj Udligning i Headphone Applikation

Indlejret Signal Behandling - Projektoplæg

Gruppe 1

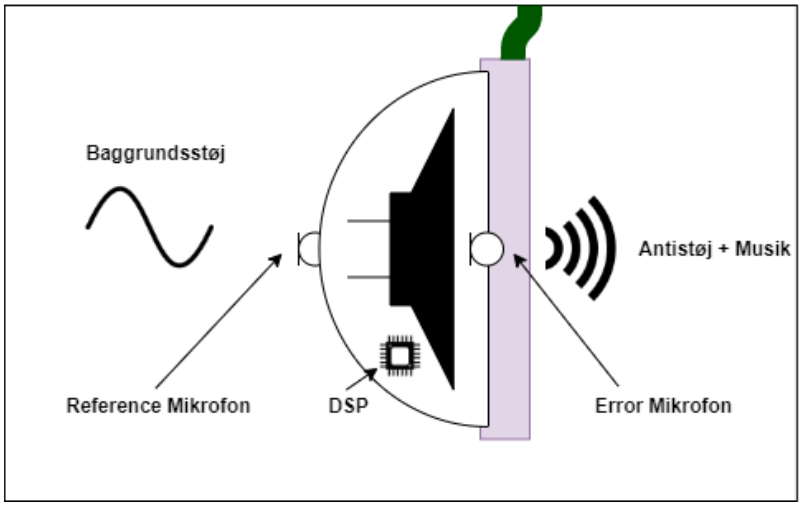
Navn	Studienummer	Retning
Mathias Ørnstrup Hvidberg	201905706	E
Niels Højrup Pedersen	201604812	E
Jakob Saugbjerg Lange	201907544	E

Indhold

1.0 Introduktion og Problembeskrivelse

I denne rapport gennemgås gruppe 1's forslag til ISB-eksamensprojektet, og det dertilhørende forarbejde der er lavet. Først introduceres og specificeres projektet ud fra en række funktionelle/ikke-funktionelle krav. Herefter gennemgås hvilken signalbehandlingsteori, der skal anvendes for at kunne implementere

Projektet går ud på at implementere aktiv støjudligning (ANC) til brug i høretelefonsapplikationer. ANC-høretelefoner er efterhånden blevet meget populære på consumer audiomarkedet, og kan ses implementeret i både in-ear og over-ear produkter med lang batterilevetid og imponerende dæmpning. De fleste kommercielle produkter leverer mellem 25-35 dB i det relevante frekvensområde (typisk 20-2000 Hz). De højere frekvenser håndteres med god isolering vha. skum og lignende materialer. Et typisk ANC-system ser ud som illustreret nedenfor på Figur 1.



Figur 1 - Illustration af et ANC-system anvendt i høretelefoner

I et over-ear system fastmonteres der et sæt mikrofoner på selve højtalerkoppen. Den første, benævnt "reference mikrofon" anvendes til at optage baggrundsstøjen. Dette signal føres ind i en DSP, som på baggrund af støjsignalet genererer et "antistøjssignal", der spilles ud af højtaleren. Dette signal er ideelt set bare støjsignalet inverteret, hvilket giver anledning til destruktiv interferens mellem støj og antistøj, heraf navnet "støjudligning". En række yderligere faktorer spiller også ind i generering af signalet, hvorfor der også sidder en "error mikrofon", men mere herom senere.

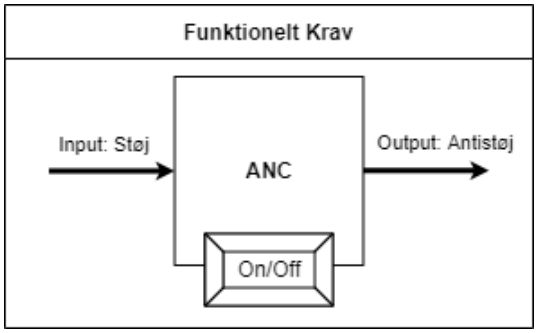
Projektet vil primært fokusere på selve implementeringen af den type filtrer og det software, som skal til for at løse problemet, så optimalt som overhovedet muligt, da selve den mekaniske udformning af headsettet samt den dertilhørende akustiske analyse ligger uden for fagets fokusområde. Dog vil der blive lavet et "mock-up" setup til at foretage funktionelle tests, foruden teoretiske verificeringer i MATLAB. Koden implementeres på Analog Devices platformen BF533.

2.0 Krav

I dette afsnit gennemgås de overordnede krav til systemet, dels funktionelle og ikke-funktionelle.

2.1 Funktionelle krav

Da systemet har minimal brugergrænseflade foruden en on/off knap, er det valgt ikke at specificere use-cases. Funktionaliteten kan illustreres helt enkelt som set nedenfor:



Figur 2 - Funktionaliteten for systemet

På den baggrund kan man opstille følgende funktionelle krav:

- Systemet skal dæmpe baggrundsstøj vha. aktiv støj udligning.
- Systemet skal have en on/off knap, der skifter mellem ANC mode og normal mode.
- Systemet skal kunne håndtere forskellige typer støjkilder i frekvensområdet 100-2000 Hz.
- Systemet skal adaptivt regulere sit antistøj-signal, og kunne håndtere ændringer i det akustiske miljø.

2.2 Ikke-funktionelle krav

Nedenfor fremgår systemets ikke-funktionelle krav, hvilket dækker over de objektive kriterier som systemet skal overholde.

Performance

- R1: Systemet skal sikre minimum 20 dB dæmpning af baggrundsstøj målt fra brugerens øre.
- R2: Systemet skal kunne dæmpe baggrundsstøj i frekvensområdet 100 Hz - 2000 Hz.
- R3: Systemet skal have en latenstid fra input til output på maksimalt 0.20 ms pr. sample[1]
- R4: Systemet skal have et dynamikområde på 96 dB (16-bit opløsning).
- R5: Systemet skal tilpasse sig et nyt støjmiljø på maksimalt 10 ms.
- R6: Systemet må bruge maximum 90% af BF533's ydeevne

Effektivitet

- R7: Systemet skal kunne adapteres til batteridrevne applikationer og må maksimalt trække 70 mW (20mA, 3.3V) i filtermode.
- R8: Systemets softwareapplikation skal maksimalt fylde 1 kB.
- R9: Systemets filtrer må maksimalt anvende 10kB memory.
- R10: Systemets algoritme skal realiseres med fixed point aritmetik.

[1] Se Appendix afsnit A.

2.3 Afledte krav

- DR1: Systemet skal sample med 48kHz.
- DR3: Systemets algoritme må maksimalt bruge 120.000 cycles pr. sample.

3.0 Teori

I dette afsnit gennemgås den teori der skal anvendes for at kunne implementere et ANC-system.

3.1 LMS

Hele ANC-systemet bygger på LMS-algoritmen[1] (Least-mean-square), som tilpasser filtervægte således at differencen mellem filterets output-signal, $y(n)$, og et ønsket signal, $d(n)$, går mod nul. Denne difference kaldes error-signalet, $e(n)$.

$$e(n) = d(n) - y(n) \tag{1}$$

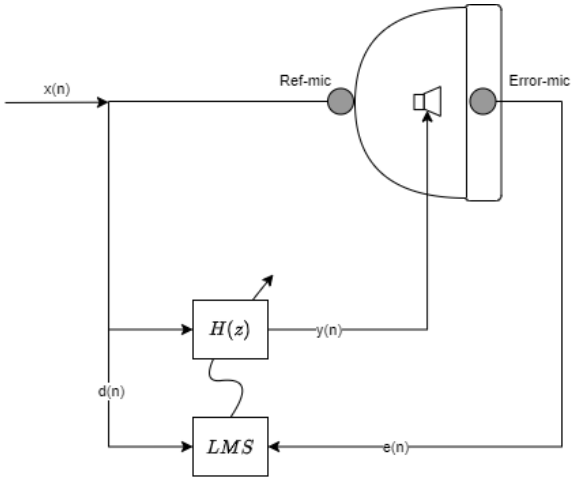
Filteret er et simpelt M-tap FIR-filter, se ligning (2).

$$y(n) = \sum_{m=0}^{M} w_m(n)x(n-m) \tag{2}$$

Selve tilpasningen af vægtene sker ved ligning *(3)*.

$$\boldsymbol{w_m}(n+1) = \boldsymbol{w_m}(n) + 2\mu e(n)\boldsymbol{x}(n) \tag{3}$$

Hvor μ repræsenterer en step-size, dvs. en faktor der afgør hvor hurtigt vægtenes størrelse justerer sig mod deres konvergensniveau. Den kan dog også blive så stor, at fejlen vil begynde at oscillere fra en yderlighed til en anden, så derfor skal den bestemmes nøje. Signalgrafen for et typisk ANC headset system ser ud som set på Figur 3 (Simpel form).

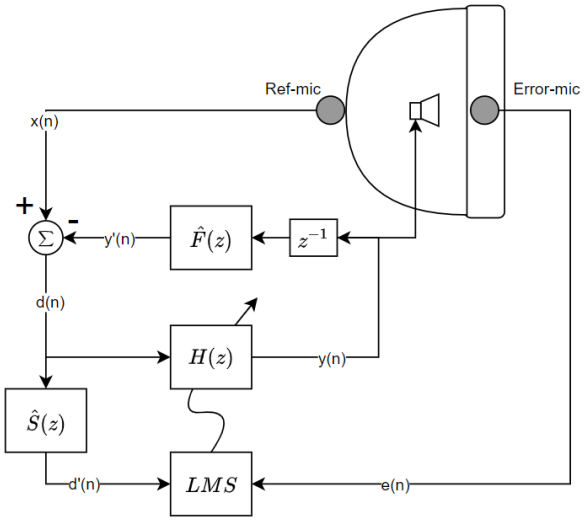


Figur 3 - Signalgraf for simplificeret ANC-system

LMS-algoritmen justerer filtervægte for filteret $H(z)$ for at generere et antistøj-signal ud af højtaleren. Som input tager den lyd fra både reference og error mikrofonen. Lyden fra referencemikrofonen repræsenterer signalet $d(n)$ fra Eq. 1, altså støjsignalet, der skal udlignes med et signal $y(n)$, som outputtets fra filteret. Error-mikrofonen levere signalet $e(n)$, som skal tilnærme sig et minimum. Denne fremstilling repræsenterer dog ikke helt virkeligheden, hvorfor der introduceres Fx-LMS.

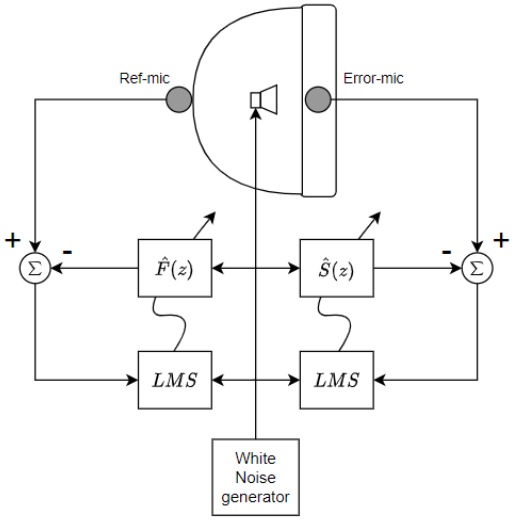
3.2 Fx-LMS

Med Fx-LMS[2] tages der højde for såkaldte "secondary paths", som dels dækker over den akustiske feedback, som opstår mellem højtaleren og de to mikroner, og dels repræsenterer den overføringskarakteristik, som konverteringerne mellem analoge, digitale og akustiske signaler har. Dette er nødvendigt for at undgå tilfælde, hvor et feedback forsager ustabile oscilleringer. Dette gøres ved at lave en slags system-kalibrering, hvor man vha. LMS-filtrer identificerer disse overføringskarakteristikker, og derefter indsætter deres koefficienter i signalvejene, som set på figuren nedenfor (inspireret af [3]):



Figur 4 - Signalgraf for ANC-systemet

Hvor funktionerne $\hat{S}(z)$ og $\hat{F}(z)$ er estimater af de overføringskarakteristikker. De bestemmes ved at afspille et hvidstøjssignal ud af højtaleren og lave to systemidentifikationer med LMS-filtrer forbundet til hver mikrofonudgang. Opstillingen kan ses nedenfor:



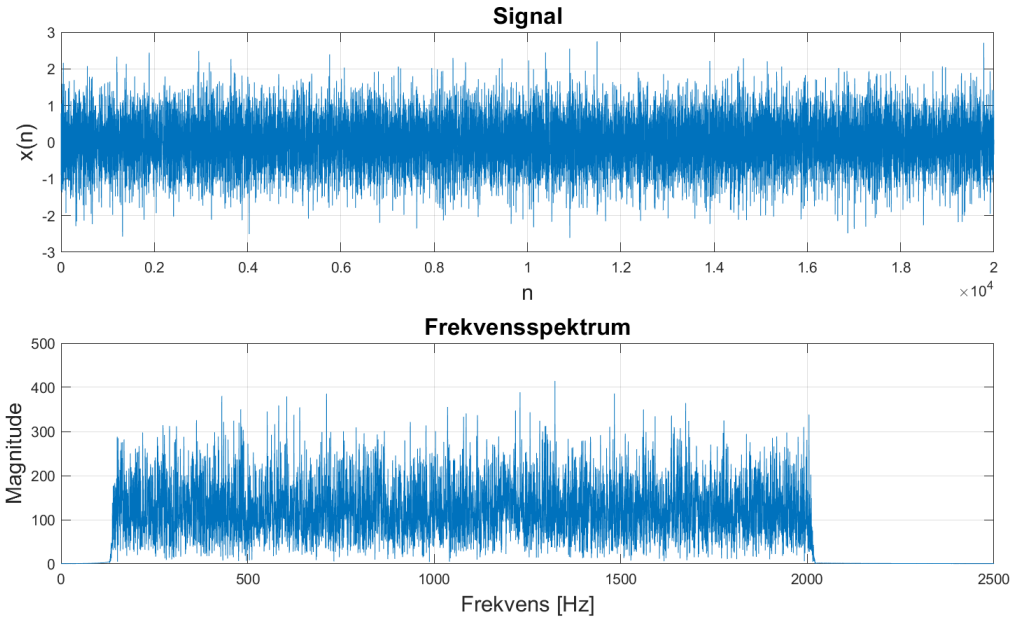
Figur 5 - Signalgraf for Path-estimering

4.0 MATLAB Løsning

I det følgende gennemgås matlab implementeringen inspireret af [4] og [5].

4.1 Path-estimation

For at estimere secondary paths genereres der et hvidstøjssignal i frekvensspektret 100Hz til 2000Hz. I matlab gøres dette med en randn-funktion, hvorefter der sættes et båndpasfilter på, se Figur 6. Dette støjsignal afspilles på højtaleren i headsettet og optages med både error-mikrofonen og reference-mikrofonen, for at kunne estimere hhv. $\hat{S}(z)$ og $\hat{F}(z)$, se Figur 5.



Figur 6 - Det genererede hvidstøjssignal til estimering af secondary paths

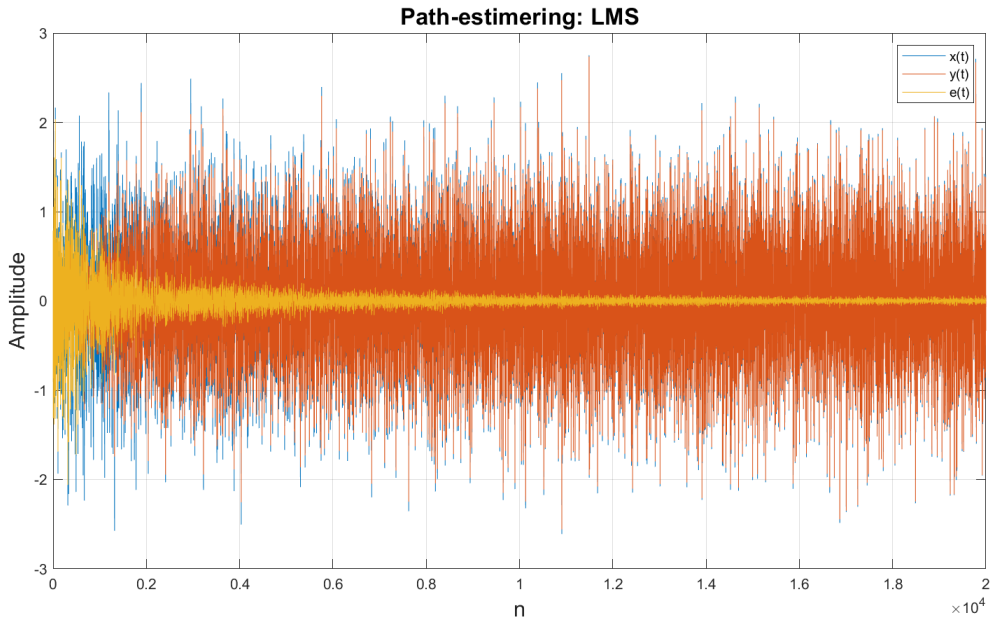
LMS-algoritmen er implementeret som en funktion i matlab, se Listing 1. Funktionen udfører først filtreringen af input-signalet med vægtene initieret til 0, hvorefter error udregnes, og til sidst adapterer funktionen vægtene. Dvs. hhv. formlerne fra teorien (2), (1) og *(3)*.

```
function [y, e, w] = LMS(x ,d ,N, M, mu)
    % LMS takes 5 parameters:
    %   x:  Input signal
    %   d:  Desired signal
    %   N:  Number of samples
    %   M:  Number of filter-taps
    %   mu: Adaption stepsize
    %
    % And returns 3 vectors [y, e, w]
    %   y: Filtered signal
    %   e: Error signal
    %   w: Filter-weights

    w = zeros(1,M); % Weights
    y = zeros(1,N); % Filtreret signal
    e = zeros(1,N); % error-signal
    for n=1:N

        % Filtering
        for m=1:M
```

```
            if (n > m)
                y(n) = y(n) + w(m)*x(n-m+1);
            end
        end

        % Error calculation
        e(n) = d(n) - y(n);

        % Weight adaption
        for m=1:M
            if (n > m)
                w(m) = w(m) + mu*e(n)*x(n-m);
            end
        end

    end

end
```

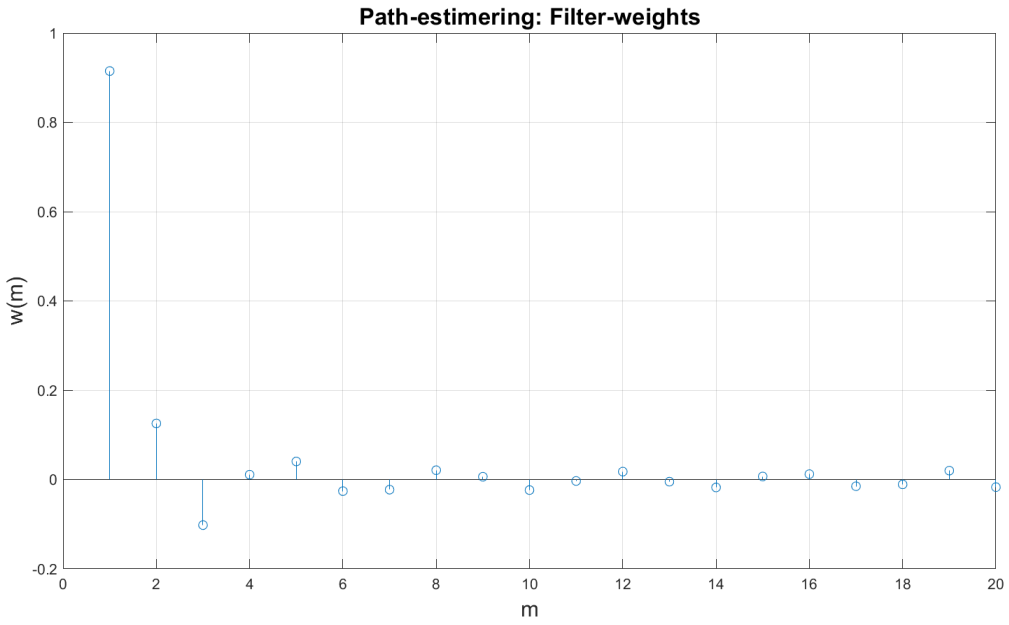
Listing 1 - LMS-funktion i matlab

Formålet er som sagt genskabe den hvide støj med LMS-algoritmen, således filter-vægtene simulerer det secondary path, $S(z)$, og det akustiske feedback path $F(z)$. De to filtre som dannes kaldes $\hat{S}(z)$ og $\hat{F}(z)$. På Figur 7 ses et plot af LMS-funktionens outputs, error-signalet og det filtrerede signal, sammen med input-signal. Filteret har 20 taps og LMS benytter en stepstørrelse, μ, på 0.002.

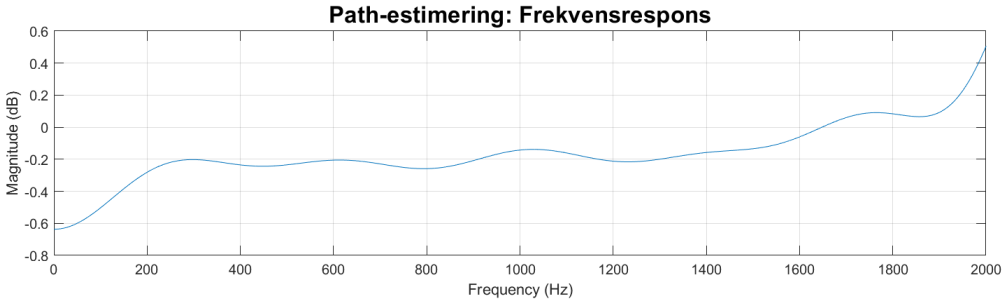


Figur 7 - Plot af input-signal, filtrerede signal og error-signalet fra LMS til path-estimering

Filtervægtene er plottet på Figur 8 og en frekvensrespons af filteret kan ses på Figur 9. Denne estimerings-metode af overføringskarakteristikkerne, $\hat{S}(z)$ og $\hat{F}(z)$, er ens for begge, som vist på Figur 5.



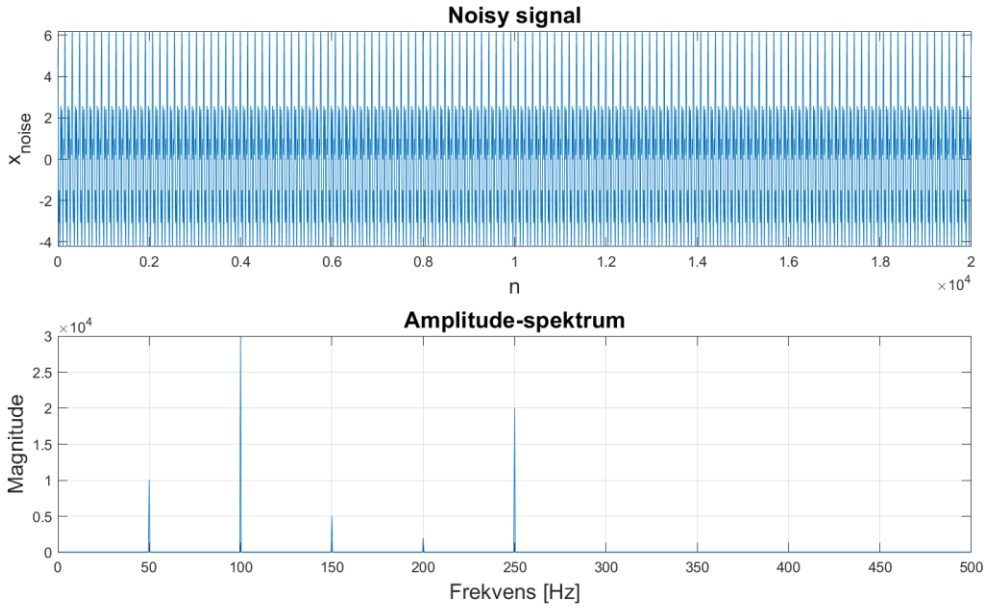
Figur 8 - Filter-vægtene til path-estimeringen



Figur 9 - Frekvensrespons af path-estimerings filter

4.2 ANC – Primary path estimation

På Figur 10 ses et genereret støjsignal. Signalet består af nogle 50 Hz, 100 Hz, 150 Hz og 250 Hz sinuskomponenter, som også fremgår af den nederste del af Figur 6.



Figur 10 - Testsignal som består af 50 Hz, 100 Hz, 150 Hz og 250 Hz sinuskurver.

På Listing 2 ses koden, der bliver brugt til at lave testsignalet.

```
%% ANC: Støjsignalet

A     = [1, 3, 0.5, 0.2, 2];
N_A   = length(A);
f0    = 100;
n_A   = 1:N_A;
f     = f0*n_A;
phase = rand(1,N_A); % Random initial phase
t     = [0:N-1]/fs;

x_noise = A(1)*cos(2*pi*t*f(1)+phase(1))
          + A(2)*cos(2*pi*t*f(2)+phase(2))
          + A(3)*cos(2*pi*t*f(3)+phase(3))
          + A(4)*cos(2*pi*t*f(4)+phase(4))
          + A(5)*cos(2*pi*t*f(5)+phase(5));

X_noise = fft(x_noise, N);
```

Listing 2 - Matlab-kode til generering af støjsignal

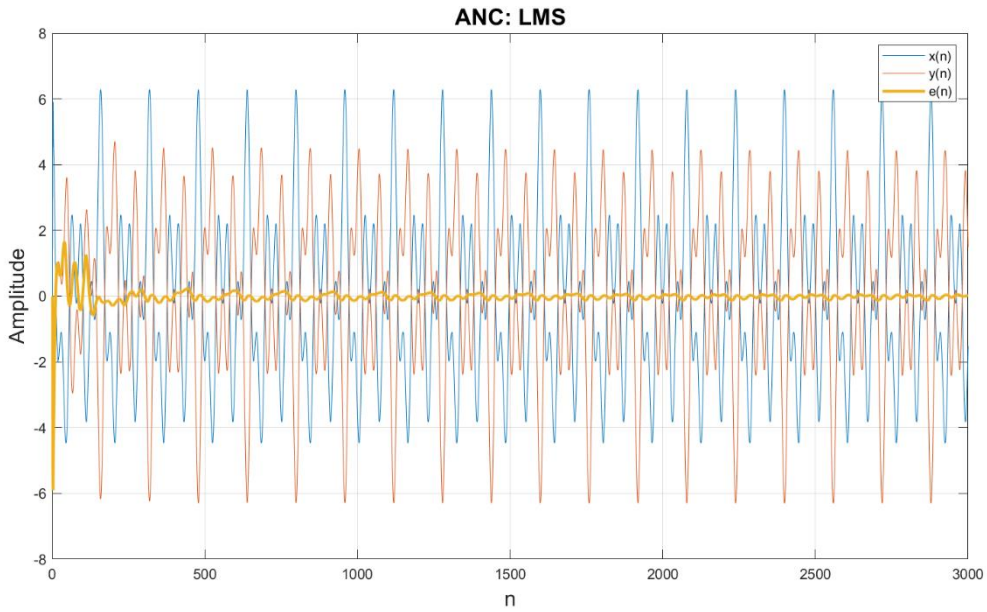
På Figur 11 ses inputsignalet x(n), det filtreret signal y(n) og den samlet fejl e(n). Formålet er at estimeret y(n) til at være -x(n), således at e(n) ≈ 0.

$$e(n) = -x(n) - y(n)$$

Ved at få y(n) til at være -x(n), vil signalerne lave destruktiv interferens. LMS-filteret som ses på Figur 11, bruger følgende $m_{filtertaps}$ og en μ:

$$m_{filtertaps} = 50 \qquad \mu = 0.001$$

På Figur 11 ses hvordan filteret skal bruge ~150 samples for at få estimeret y(n) korrekt. Efter ~150 samples er signalet estimeret så godt, at e(n) tilnærmer sig ~0.



Figur 11 - input-signal x(n), filtreret-signal y(n) og fejl-signal e(n).

5.0 Prototype

På Figur 12 ses prototypen til projektet. Til error og reference-mikrofon er der valgt BOB-12758-boards[6]. Disse mikrofon-boards, har en forforstærker som forstærker mikrofonsignalet til et passende spændingsniveau. De to mikrofon-boards, bliver forsynet af en ekstern 5V strømforsyning. Error og reference-mikrofonerne er placeret på henholdsvis inde og ydersiden af højtalerkoppen. Outputtet fra reference og error-mikrofonen bliver sendt igennem et højpas-filter, for at fjerne DC-komponenten af signalet, inden det bliver samplet af DSP'en. Lydsignalet til højtaleren bliver leveret igennem et AUX-kabel. På Figur 13 ses placeringen af de to mikrofoner.

Figur 12 - De samlede elementer som opgør prototypen for projektet.


Figur 13 - På venstre billede ses placeringen af reference-mikrofonen. På højre billede ses placeringen af error-mikrofonen.

6.0 Matlab-kode

```
clc; clear; close all;
%% Konstanter til plotvinduesstørrelser
x0=10;
y0=10;
width=1100;
height=600;

%% Path-estimering: White noise generator
fs = 8000;              % Samplingsfrekvens
N = 20000;              % Antal samples
n = 1:N;                % Sample
f_res = fs/N;           % Frekvensopløsning
f_axis = [0:N-1]*f_res; % Frekvensakse

% Generer hvid støj mellem 150Hz og 2000Hz
x = randn(1,N);
x = bandpass(x,[150,2000],fs);

% Fast-fourier-transformation af støjen
X = fft(x, N);

% Plot af signalet og dets frekvensspektrum
figure()
subplot(2,1,1)
plot(n,x)
grid on;
xlabel('n','FontSize', 15);
ylabel('x(n)','FontSize', 15);
title('Signal','FontSize', 16);
subplot(2,1,2)
plot(f_axis(1:end/2), abs(X(1:end/2)))
```

```
grid on;
xlabel('Frekvens [Hz]','FontSize', 15)
ylabel('Magnitude','FontSize', 15)
title('Frekvensspektrum','FontSize', 16)
xlim([0 2500])
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'White_nose_generator.png');

%% Path-estimering: LMS
M = 20;         % et M-tap filter
mu = 0.002;      % stepsize
help LMS
[y, e, w] = LMS(x, x, N, M, mu);

%% Path-estimering: Plot af LMS
n = 1:N;
figure()
plot(n,x)
hold on;
plot(n,y)
plot(n,e)
legend('x(t)', 'y(t)', 'e(t)')
grid on;
xlabel('n','FontSize', 15);
ylabel('Amplitude','FontSize', 15);
title('Path-estimering: LMS','FontSize', 16);
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'path_est_LMS.png');

%% Path-estimering: Plot af estimeret filter
figure()
stem(w)
grid on;
xlabel('m','FontSize', 15);
ylabel('w(m)','FontSize', 15);
title('Path-estimering: Filter-weights','FontSize', 16);
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'path_est_weights.png');

%% Path-estimering: Plot af frekvensresponset for estimeret filter
freqz(w,1,10000,'half',fs)
title('Path-estimering: Frekvensrespons', 'FontSize', 16)
grid on;
xlim([0 2000])
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'path_est_freqz.png');

%% ANC: Støjsignalet

A = [1, 3, 0.5, 0.2, 2];
N_A = length(A);
f0 = 50;
n_A = 1:N_A;
f = f0*n_A;
phase = rand(1,N_A); % Random initial phase
```

```
t = [0:N-1]/fs;
x_noise = A(1)*cos(2*pi*t*f(1)+phase(1)) +
A(2)*cos(2*pi*t*f(2)+phase(2)) + A(3)*cos(2*pi*t*f(3)+phase(3)) +
A(4)*cos(2*pi*t*f(4)+phase(4)) + A(5)*cos(2*pi*t*f(5)+phase(5));
X_noise = fft(x_noise, N);

figure()
subplot(2,1,1)
plot(n,x_noise)
grid on;
xlabel('n','FontSize', 15);
ylabel('x_{noise}','FontSize', 15);
title('Noisy signal','FontSize', 16);
subplot(2,1,2)
plot(f_axis(1:end/2), abs(X_noise(1:end/2)))
grid on;
xlabel('Frekvens [Hz]','FontSize', 15)
ylabel('Magnitude','FontSize', 15)
title('Amplitude-spektrum','FontSize', 16)
xlim([0 500])
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'noise_signal.png');

%% ANC: Apply path-estimate
x_noise_hat(n) = zeros(1,N);
for n=1:N
    for m=1:M
        if (n > m)
            x_noise_hat(n) = x_noise_hat(n) + w(m)*x_noise(n-m+1);
        end
    end
end
n = 1:N;
figure()
plot(n,x_noise)
hold on;
plot(n,x_noise_hat)
legend('x_{noise}(n)', 'x_{noisehat}(n)')
grid on;
xlabel('n','FontSize', 15);
ylabel('Amplitude','FontSize', 15);
title('Noisy signal filtered','FontSize', 16)
xlim([0 1000])
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'noise_signal_filtered.png');

%% ANC: LMS

M = 50;          % et M-tap filter
mu = 0.001;      % stepsize
[y, e, w] = LMS(x_noise_hat, -x_noise_hat, N, M, mu);

e_rms = rms(e(500:end))

%% ANC: Plot af LMS
```

```
n = 1:N;
figure()
plot(n,x_noise_hat)
hold on;
plot(n,y)
plot(n,e, 'LineWidth',2)
legend('x(n)', 'y(n)', 'e(n)')
grid on;
xlabel('n','FontSize', 15);
ylabel('Amplitude','FontSize', 15);
title('ANC: LMS','FontSize', 16);
xlim([0 3000])
set(gcf,'position',[x0,y0,width,height])
saveas(gcf,'ANC_LMS.png');
```

7.0 Appendix

7.1 A)

Afstand fra reference-mikrofon til trommehinden:

$$a = 5cm + 2.5cm = 7.5cm$$

Lydens hastighed:

$$v_{lyd} = 343 \cdot \frac{m}{s}$$

Tid det tager lyden at komme fra reference-mikrofon til trommehinden

$$t = \frac{a}{v_{lyd}} = 0.219\, ms$$

BF533 har en clockfrekvens på 600 MHz. Hvis algoritmen maxmimum må belaste DSP'en med 90%, svarer det til en clockfrekvens på:

$$f_{BF533} = 600MHz \cdot 0.9 = 540\, MHz$$

Én clock-cycle vil derfor tage:

$$T_{BF533} = \frac{1}{f_{BF533}} = 1.852\, ns$$

Antallet af clock-cycle BF533 kan nå i tiden *t* bliver dermed:

$$clock_{cycle} = \frac{time}{T_{BF533}} \approx 1.181 \cdot 10^5$$

Dette er dermed det maksimale antal clockcycles der må benyttes fra input til output af filteret.

Litteraturliste

[1] W.-S. Gan and S. M. Kuo, *Embedded Signal Processing with Micro Signal Architecture*. Wiley.

[2] B. Farhang-Boroujeny, *Adaptive Filters. Theory and Applications*, Second Edi. WILEY.

[3] K. C. Chen, C. Y. Chang, and S. M. Kuo, "Active noise control in a duct to cancel broadband noise," in *IOP Conference Series: Materials Science and Engineering*, Sep. 2017, vol. 237, no. 1. doi: 10.1088/1757-899X/237/1/012015.

[4] "Active Noise Control Using a Filtered-X LMS FIR Adaptive Filter - MATLAB & Simulink - MathWorks Nordic." https://se.mathworks.com/help/audio/ug/active-noise-control-using-a-filtered-x-lms-fir-adaptive-filter.html (accessed Apr. 22, 2022).

[5] K. C. Chen, C. Y. Chang, and S. M. Kuo, "Active noise control in a duct to cancel broadband noise," in *IOP Conference Series: Materials Science and Engineering*, Sep. 2017, vol. 237, no. 1. doi: 10.1088/1757-899X/237/1/012015.

[6] Sparkfun, "SparkFun Electret Microphone Breakout - BOB-12758 - SparkFun Electronics." https://www.sparkfun.com/products/12758 (accessed Apr. 18, 2022).