Jeppe Skødt Benjaminsen 201500154

Søren Kruse Holm 201409556

Mikkel Poulsen 20112893

Mathias Vandel 201409554

DSB miniprojekt - Equalizer

# Indledning

Vores equalizer er lavet med 5 forskellige pasbånd ligeligt opdelt fra 0-20 kHz, og er implementeret med FIR-filtre.

Lavpasfilter = <4 kHz  
Båndpas1 = 4-8 kHz  
Båndpas2 = 8-12 kHz  
Båndpas3 = 12-16 kHz  
Højpasfilter = >16 kHz

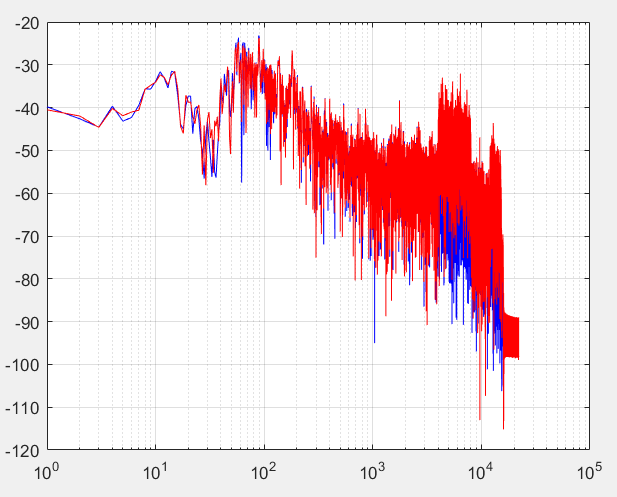
Nedenfor i testen vil vi referere til parametrene med en 5 ciffer lang liste, hvor de angiver forstærkningen fra lavpas til højpas. Fx betyder (1,1,1,4,4) at de første 3 bånd er forstærket med 1 og de to sidste med 4 (svarende til 12 dB).

# Test af equalizer

Her kan det ses hvordan forstærkning/dæmpning af de forskellige bånd giver sig til udtryk i frekvensdomænet.

På y-aksen er amplitude i dB, på x-aksen er frekvens. Det blå signal er det originale signal, det røde er efter equalizeren er påført.

Her på Figur 1 er forstærkningerne i equalizeren angivet til (1,4,1,4,4).



Figur

# Matlab kode

## Båndpasfilter

Her ses koden fra ét af båndpasfiltrene. Koden fra de forskellige filtre er ens bortset fra knækfrekvenserne. De er fremstillet med FDA-tool. Filter-funktionerne returnerer filterkoefficienterne.

function b = BP\_8\_12

%BP\_8\_12 Returns a discrete-time filter object.

% MATLAB Code

% Generated by MATLAB(R) 8.6 and the Signal Processing Toolbox 7.1.

% Generated on: 27-Apr-2016 11:08:10

% FIR Window Bandpass filter designed using the FIR1 function.

% All frequency values are in Hz.

Fs = 44100; % Sampling Frequency

N = 1000; % Order

Fc1 = 8000; % First Cutoff Frequency

Fc2 = 12000; % Second Cutoff Frequency

flag = 'scale'; % Sampling Flag

% Create the window vector for the design algorithm.

win = hann(N+1);

% Calculate the coefficients using the FIR1 function.

b = fir1(N, [Fc1 Fc2]/(Fs/2), 'bandpass', win, flag);

%Hd2 = dfilt.dffir(b);

% [EOF]

end

## Lavpasfilter (vinduesmetoden)

Denne implementering af lavpasfiltret virker i sig selv, men vi havde problemer med at få den til at virke som funktion, så den kunne være en del af den samlede equalizer. Vi ønskede alligevel at den skulle med i rapporten.

clear; clc

[x,f\_sample] = audioread('metallica.wav'); %load lyd

f\_cutoff = 4000;

M = 3000;

freq\_resolution = f\_sample / M;

k\_cutoff = f\_cutoff / freq\_resolution;

k\_cutoff\_int = round(k\_cutoff);

k\_cutoff\_int = round(k\_cutoff)

disp(['svarer til ' num2str(k\_cutoff\_int\*freq\_resolution) ' Hertz'])

H\_left = [1 ones(1,k\_cutoff\_int) zeros(1,(M/2)-k\_cutoff\_int) zeros(1,M/2)];

H\_right = [0 zeros(1,M/2) zeros(1,(M/2)-k\_cutoff\_int) ones(1,k\_cutoff\_int)];

H = H\_left + H\_right;

h = fftshift(real(ifft(H))); %filter uden hanning

w\_hanning = hanning(M+1)'; %definering af hanning vindue

h\_win = h.\*w\_hanning; %filter med hanning

w\_hanning = rot90(w\_hanning);

H\_with\_win = fft(h\_win,f\_sample);

x\_fft = fft(x,f\_sample);

Y1 = filter(h,1,x\_fft);

H\_with\_win = rot90(H\_with\_win);

Y2 = Y1.\*H\_with\_win;

N = length(Y2);

delta\_f = f\_sample/N;

f\_axis = [0:delta\_f:f\_sample-delta\_f];

figure(1)

semilogx(f\_axis(1:0.5\*end), 20\*log10(abs((2/N)\*Y2(1:0.5\*end))));

figure(2)

plot(abs(H\_with\_win(1:f\_sample/2)),'r','linewidth',2)

## Equalizer funktion

Her ses koden til selve vores equalizer funktion. Den tager 5 parametre, som angiver forstærkningen på de 5 forskellige bånd.

function output = equalizer(lp,bp1,bp2,bp3,hp)

[x f\_sample] = audioread('angerfist.wav');

h = @BandPass12\_16;

x1 = h();

Y = filter(x1,1,x);

h2 = @bandpass40008000;

x2 = h2();

Y2 = filter(x2,1,x);

h3 = @BP\_8\_12;

x3 = h3();

Y3 = filter(x3,1,x);

h4 = @hp\_filter;

x4 = h4();

Y4 = filter(x4,1,x);

h5 = @lp\_filter;

x5 = h5();

Y5 = filter(x5,1,x);

output = Y5\*(lp) + Y4\*(hp) + Y3\*(bp2) + Y2\*(bp1) + Y\*(bp3);

end