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).

For at finde ud af hvor mange gange der skulle multipleres, brugte vi følgende ligning .

# Teori

Dette afsnit beskriver teorien bag noget af det essentielle for at løse denne opgave.

## Digitale filtre

Dettte miniprojekt baseret på digitale filtre som består af to typer: FIR (Finite impulse response) og IIR (Infinite impulse response). Et FIR filter har definitionen:

Hvor y(n) er outputtet, M er filterets længde og bk er filterets koefficienter. I matlab er der en indbygget funktion, der hedder filter(b,a,x) hvor b er filterets koefficienter, a er filterets feedback koefficinter og x er det signal, der ønskes filtreret. Ved et FIR filter vil filterets feedback koefficienter a, altid være 1, da et FIR filter ingen feedback har. Dette er også forskellen fra et IIR filter, hvis vi ser på definitionen for IIR filteret:

Som der ses er der kommet et led mere på, som er et feedback. I matlab kan samme funktion filter funktion bruges, men nu er filterets feedback koefficienter a, ikke længere 1 men en vector der bestemmer feedbacken.

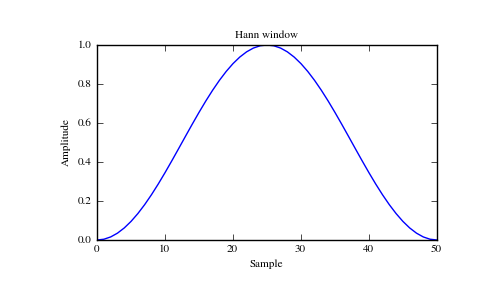
Fordelen ved et IIR filter fremfor et FIR filter er at der er færre koefficienter og dermed færre beregninger, altså kræver filteret ikke så meget. Til gengæld er et FIR filter mere effektivt, og også nemmere at implementere.

I dette projekt er der kun benyttet FIR filtre, da disse er simplere at implementere og har den funktionalitet, der ønskes for at løse opgaven.

## Hanning

Vores filtre er alle designet med et vindue. Der er valgt et hanning vindue, der har forskriften:

Hvilket resulterer i et vindue, der ser ud som på Figur 1. Vinduet ganges så på filteret i frekvensdomænet. Hermed opnås et bedre filter, da vinduet vil dæmpe ripples i yderområderne af filteret, hvor der typisk er ripples. Dog har det den påvirkning på filteret at det ikke er helt så skarpt, men det opvejes af dens anden funktionalitet.

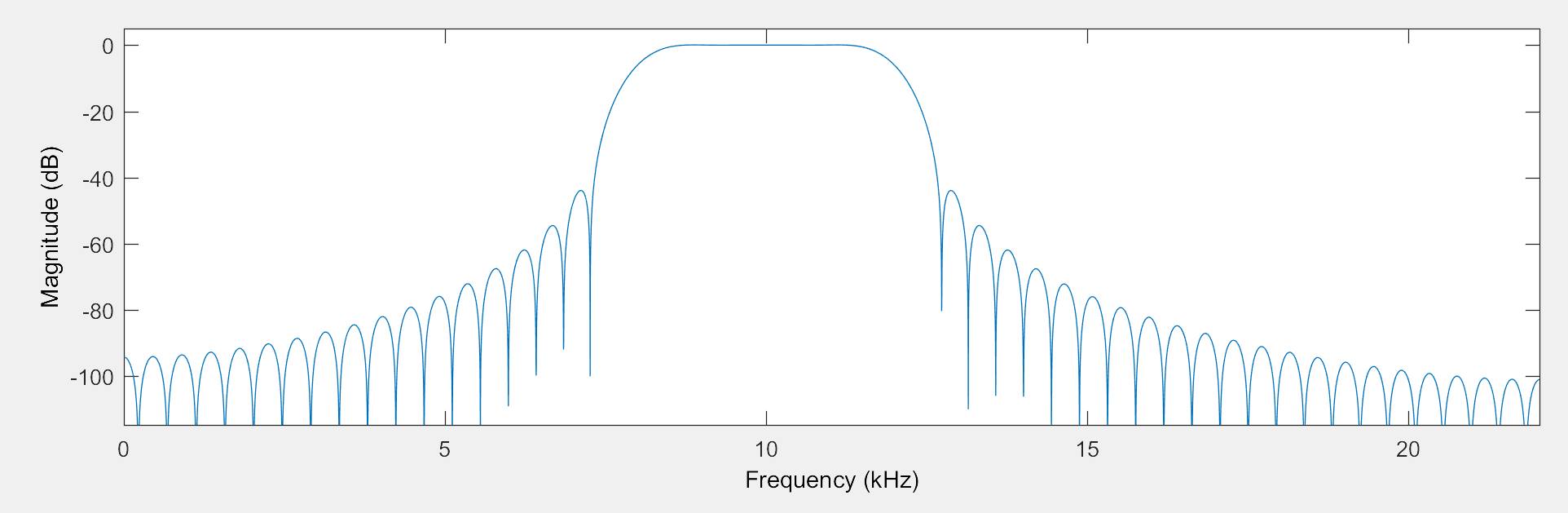


Figur 1 Hanning vindue

# Design af filter

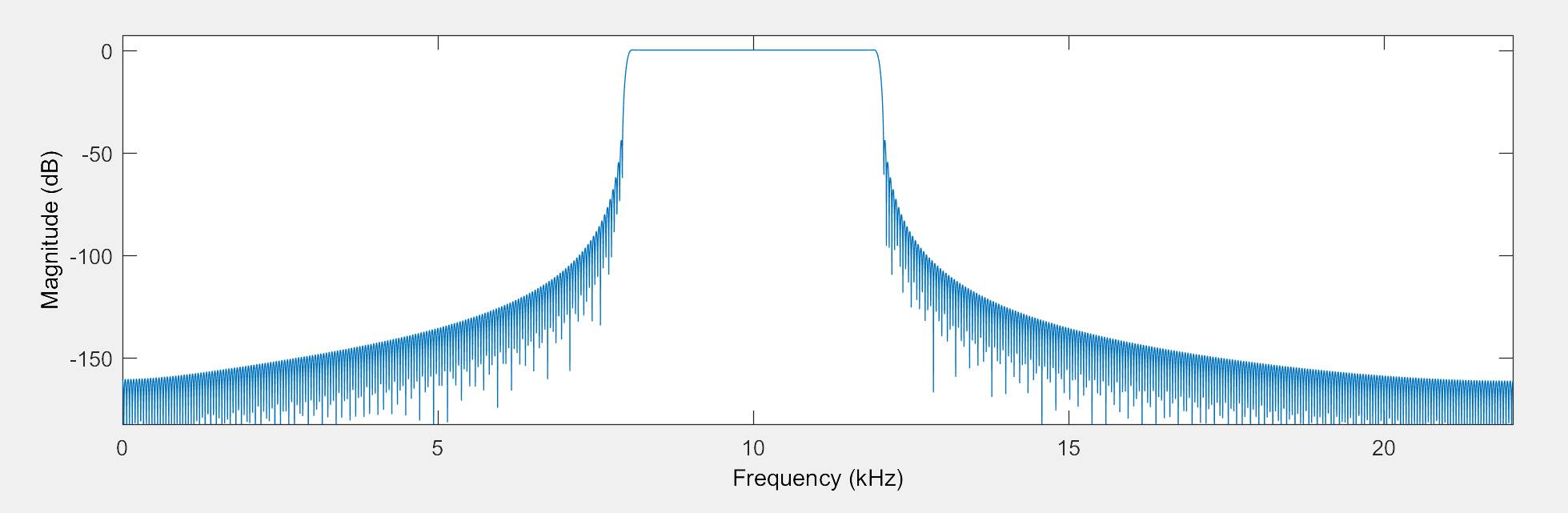
I opgaven blev der eksperimenteret med filterorden. Ved lav filterorden dæmpes dB’en betydeligt langsommere og har færre ripples.

Figur 1 viser et båndpassfilter fra 8-12 kHz, med en filterorden på 100.



Figur

Figur 2 viser et filter med en filterorden på 1000. Her dæmpes med 50db, før der begynder at komme ripples, hvilken resulterer i et bedre filter.



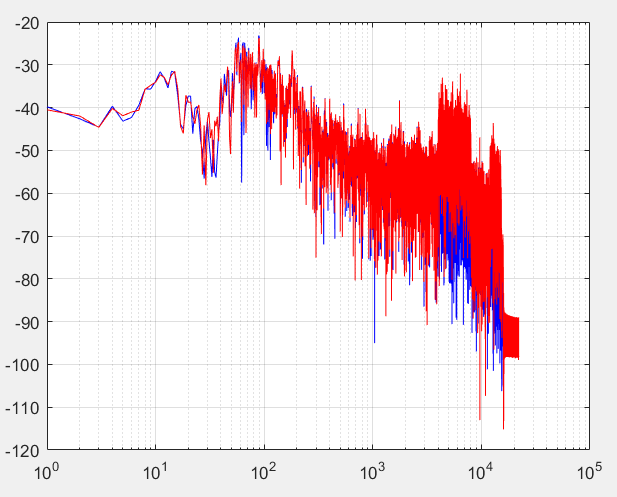
Figur

# Test af equalizer

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

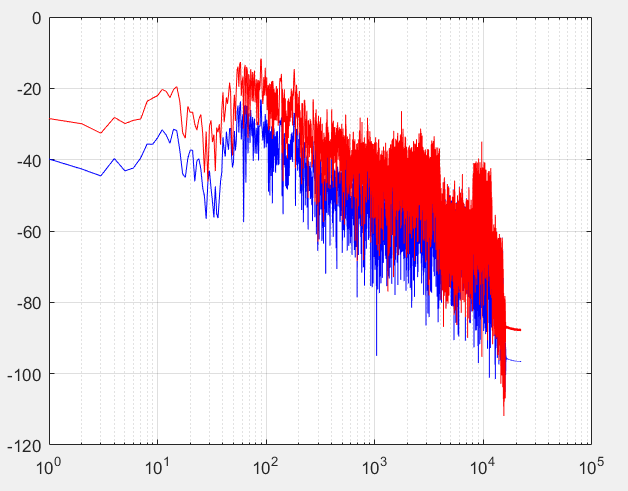
På y-aksen er amplitude i dB, på x-aksen er frekvens. Det blå signal er det originale signal, det røde er efter equalizeren er påført.

Her på Figur 3 er forstærkningerne i equalizeren angivet til (1,4,1,4,4).



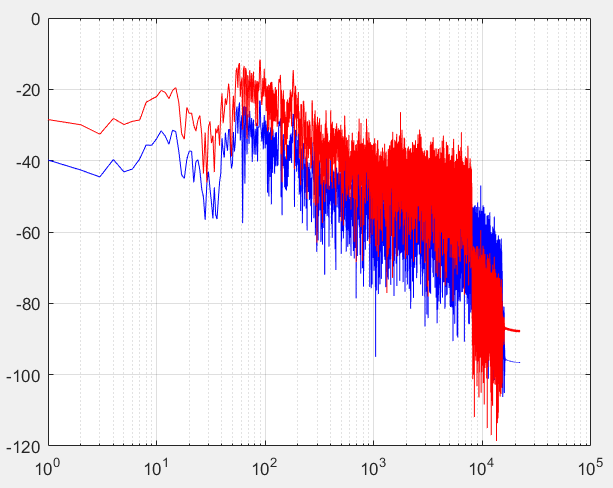
Figur

Figur 4 viser en equalizer hvor forstærkningen er angivet til (4, 4, 1 ,4 , 0.25)



Figur

På Figur 5 har vi prøvet at dæmpe det høje frekvenser. Equalizeren er sat til (4, 4, 0.25, 0.25, 0.25)



Figur

Dette resultede, et bass dominerende output.

# Impulsresponser

Her tager vi et kig på impulsresponser for vores bandpass filter. Et impulsrespons er outputtet fra et filter, når det bliver udsat for en kort input signal. Dette kunne for eksempel være hvis en bil, pludseligt kører i et hul. Her ville man kunne betragte støddæmperens reaktion som en impulsrespons.

I Matlab har vi anvendt værktøjet *fdatool* til at lave vores bandpass filter. For at kunne lave et fint billede af impulsresponsen har vi anvendt et andet værktøj i Matlab som hedder *impz*, som returner impulsresponsen og sample tiden.



Figur 6 Orden: 1000, knækfrekvenser 8000 og 12000

Her ser vi et impulsrespons for vores bandpass filter. Som det kan ses, så er der et delay på ca 500 samples før impulsrespons kommer og filterets ønskede karakteristik indtræder. Forsinkelsen kan beregnes til at være 11,37 ms. (500 samples / samplefrekvens)



Figur 7 Orden: 2000, knækfrekvenser 8000 og 12000

På figur 7 er impulsresponsen for samme båndpasfilter plottet, men nu er ordenen sat op til 2000. Når man sætter ordenen op, så kommer der et større delay før filteret har den ønskede effekt. Til gengæld bliver filteret også endnu mere skarpt.



Figur 8 Orden: 1000, knækfrekvenser 10000 og 12000

På figur 8 er ordenen sat tilbage til 1000, men båndpasset går nu kun fra 10000-12000 hvor det før gik fra 8000-12000. Det kan ses at filteret er mindre skarpt end før. Vi får derfor et bedre filter hvis vi holder båndpasset på en bredde med 4000 Hz.

# 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

# Konklusion/diskussion

Vi har i denne opgave implementeret en equalizer i matlab, som lader os styre energien i 5 forskellige frekvensbånd for et signal. I den virkelige verden ville sådan en equalizer være anvendelig til at styre frekvenserne i et stykke musik, så man kan dæmpe/fremhæve bestemte instrumenter og dele af stykket.

Equalizeren består af et lavpasfilter, 3 båndpasfiltre og et højpasfilter. Disse er implementeret som FIR-filtre vha FDAtool i matlab, som nemt og effektivt lader os lave et filter, og ligeledes finjustere og putte vinduer på.

Der blev eksperimenteret med knækfrekvenser og forskellig orden, og deres indvirkning på hinanden. Hvis man fordobler ordenen fra 1000 til 2000 kunne man se på impulsresponsen, at man får et mere skarpt filter, men til gengæld fordobles delayet for filterets indvirkning også. Hvis vi ændrede et af vores båndpasfiltres passtørrelse til det halve, kunne man på impulsresponsen se, at filtret blev mindre skarpt. Derfor var en orden på 1000 og båndpasbredde på 4000 i sidste ende passende, da vi undgår for stort delay og samtidig har et skarpt filter. Man kan argumentere for, at hvis equalizeren skulle være brugt i den virkelige verden, skulle der have været flere båndpas af forskellig størrelse, så de matchede almindelige frekvensspektre for fx guitar, stortromme, bas, vokal mv. Man ville så kunne få noget øget funktionalitet, men til gengæld ville det måske gå lidt ud over filtrenes skarphed.