Jeppe Skødt Benjaminsen 201500154

Søren Kruse Holm 201409556

Mikkel Poulsen 20112893

Mathias Vandel 201409554

DSB Miniprojekt C

Indhold

[Indledning 1](#_Toc450823516)

[Teori 1](#_Toc450823517)

[Design og implementering 1](#_Toc450823518)

[Quantizer 1](#_Toc450823519)

[Ekko 6](#_Toc450823520)

[Kode til ekko-funktion 8](#_Toc450823521)

[Chorus 9](#_Toc450823522)

[Kode til chorus effekt 11](#_Toc450823523)

[Volumen 12](#_Toc450823524)

[Lydmixer implementering og eksempel på brug 13](#_Toc450823525)

[Diskussion/Konklusion 15](#_Toc450823526)

# Indledning

I dette miniprojekt har vi implementeret en lydmixer som har en række funktionaliter. Denne lydmixer har en equalizer som lader os booste og/eller dæmpe forskellige frekvensbånd i et vilkårligt stykke musik. Desuden skal det være muligt at kunne sætte ekko på musikstykket samt et chorus. Til slut skal det også være muligt at quantitize musikstykket, så man kan sænke lydkvaliteten ved at manipulere bit-raten, hvilket også resulterer i en distortion-agtig effekt.

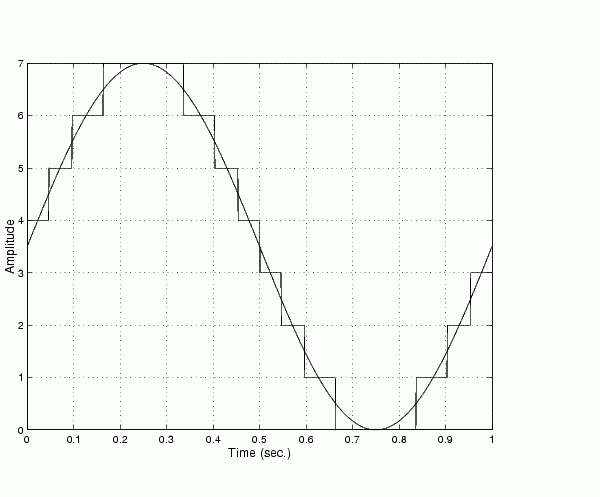
Rapporten beskriver noget af teorien bag, hvordan det er implementeret, og noget test af lydmixeren.

# Design og implementering

## Quantizer

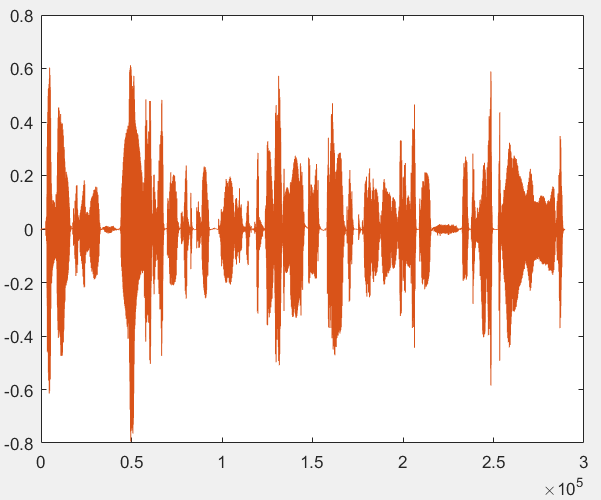
Quantizer funktionen i remixeren, har til opgave at stille på lydkvaliteten. Det er muligt at indsætte et signal der afspiller lyd ved 16bit, og reducere det til f.eks. 8bit. Dette vil resultere i en nedsat kvalitet ved afspilning af den givne lydfil, man skal dog havde en lav bit, for at høre betydelig forskel. Det er dog ikke muligt at ændre f.eks. et 16bit signal til 32bit.

På Figur 1 ses der to signaler, hvor den ene har høj bitkvalitet og den anden lav.

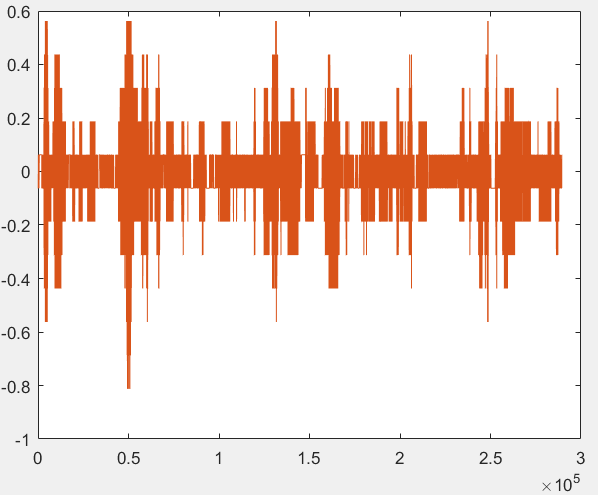


Figur 1 - høj og lav bitrate

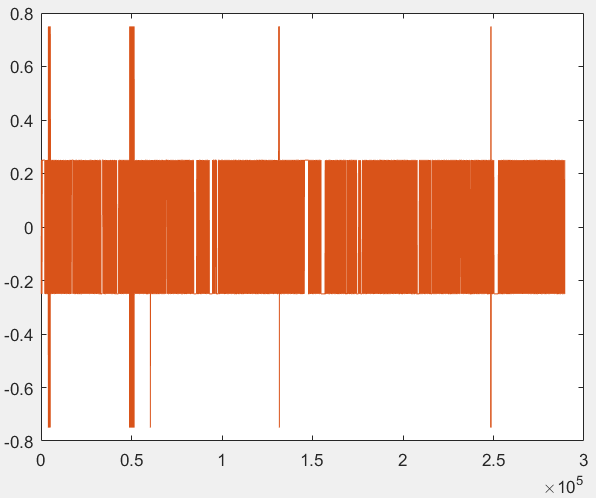
Ved en nedsænkning i kvaliteten vil der opstå en del støj. Figur 2 viser et plot diagram ved 16bit. Til sammenligning viser Figur 3 et 4bit signal og Figur 4 et 2bit.



Figur 2 plot 16bit



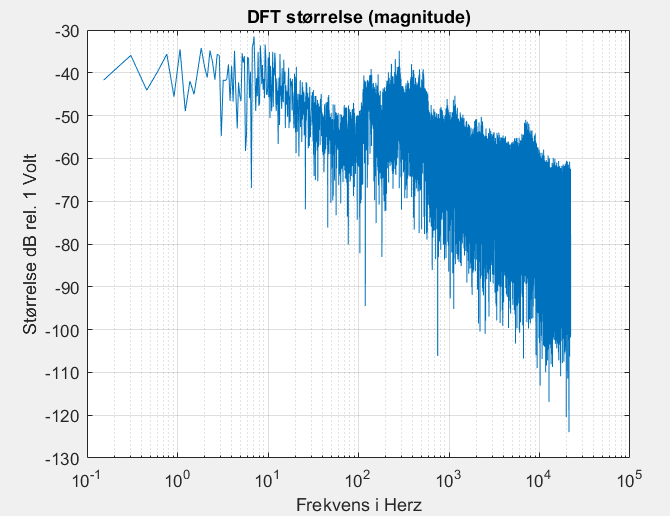
Figur 3 plot 4bit



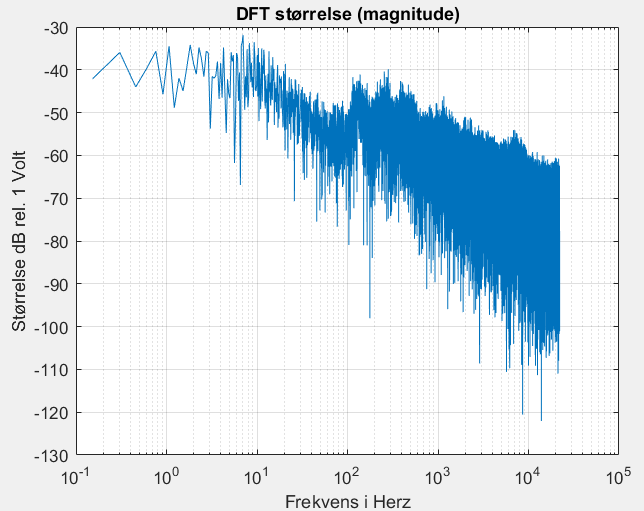
Figur 4. plot 2bit

Ved nedsætning af bitrate, ses det først visuelt ved lavere bit som 2 og 4, det samme gælder for lyden, dog kan man godt høre lidt forskel allerede ved 8bit.

Der kommer som sagt meget støj, dette ses på Figur 5, som er et 2bit DFT. For at visuallisere det lavede vi et fejl signal, og som Figur 6 viser, er fejl signalet nærmest identisk med orignale signal.

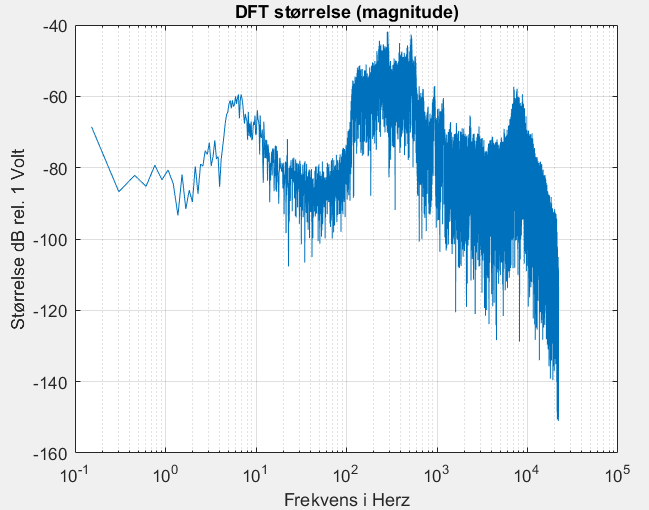


Figur 5 DFT på 2bit signal.



Figur 6 DFT på 2bit err\_signal

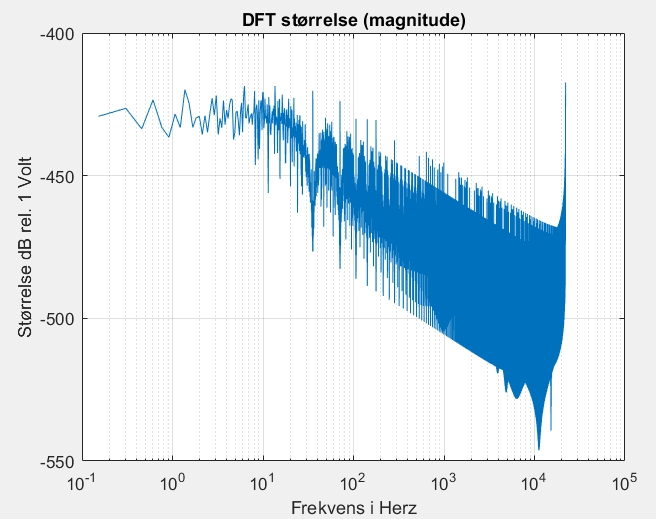
Derfor vil der forventes, at fejl signalet bliver lavere i dB ved højere bitrate, og de to signaler bliver mindre identiske. Figur 7 viser sammenlignet med 2bit signalet på Figur 5, en mere tydelig variation i dB og frekvenser.



Figur 7 DFT på 16bit signal

Som forventet ville fejl signalet ved 16bit næremst ikke eksistrere.

Figur 8 viser at dB niveauet ved 16 bit ligger omkring -450 dB, hvilket er så dæmpet, at det kan betragtes som 0, og vil ikke kunne høres.



Figur 8

## Ekko

I mixeren er der implementeret en funktion, der kan lave et ekko. Et ekko fungerer ved at den samme lyd eller signal, kommer igen efter kort tid. Ekko funktionen tager et signal, en amplitude og en delay-tid som input parametre. Amplituden siger hvor højt, det forsinkede signal er og delay-tiden siger hvor langt tid ekko-signalet er forsinket.

På Figur 9, ses x-aksen som tid og y-aksen som amplituden. Som der kan anes, er der et signal ved tiden 0 og det samme igen ved 4410. På denne måde kommer der et ekko, da signalet ved 4410 er forsinket 0,1 sekund.



Figur 9 Filterkoefficienter

Som der ses på Figur 2, er dette et kamfilter, hvilket er helt forventeligt, da et kamfilter tilføjer et forsinket version af signalet til sig selv. Det er også præcis denne funktionalitet et ekko har.



Figur 10 DFT af filter

### Kode til ekko-funktion

function s\_filtered\_voice\_ekko = ekko(s\_voice ,amplitude, ekko)

fsample = 44100;

% \*\*\*\*\* Ekko-filter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delay\_samples = round(ekko\*fsample);

b = [amplitude zeros(1,delay\_samples)];

% \*\*\*\*\* Frekvensanalyse \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

H = fft(b,fsample);

% \*\*\*\*\* Plots \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%figure(2); clf

%semilogx(20\*log10(abs(H(1:0.5\*fsample))))

%figure(1);clf

%plot(b)

% \*\*\*\*\* Filtrering på lyd \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*,

s\_voice\_left = s\_voice(:,1)'; % venstre kanal i wave-filen udtages

s\_filtered\_voice = filter(b,1,s\_voice\_left); % filtrering med b (FIR)

% \*\*\*\*\* filtreret version \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

s\_filtered\_voice\_ekko = s\_filtered\_voice/max(abs(s\_filtered\_voice)); % normalisering

%soundsc(s\_filtered\_voice\_norm, fsample)

end

## Chorus

En chorus effekt, er en videreudbygning af en ekko effekt. Ved ekkoet er der kun et delayet signal, hvor der ved en chorus effekt er flere. I dette tilfælde er effekten implementeret med 3 delays, som der også ses på Figur 3 nedenfor. Her er det oprindelige signals amplitude 1, mens de forsinkede signaler er 0,3.



Figur 11 Chrous effekt filterkoefficienter

Som der ses på Figur 4, er dette også et kamfilter. Dette giver god mening, da funktionaliteten er den samme som ved et ekko, blot med flere delay signaler.



Figur 12 DFT af filter

### Kode til chorus effekt

function s\_filtered\_voice\_norm = chorus(s\_voice, amplitude, kor1, kor2, kor3)

fsample = 44100;

% \*\*\*\*\* Chorus-filter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delay\_1 = round(kor1\*fsample);

delay2 = round(kor2\*fsample);

delay3 = round(kor3\*fsample);

b = [1 zeros(1,delay\_1) amplitude zeros(1,delay2) amplitude zeros(1,delay3) amplitude];

% \*\*\*\*\* Frekvensanalyse \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

H = fft(b,fsample);

% \*\*\*\*\* Plots \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%figure(2); clf

%semilogx(20\*log10(abs(H(1:0.5\*fsample))))

%figure(1);clf

%plot(b)

% \*\*\*\*\* Filtrering på lyd \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*,

s\_voice\_left = s\_voice(:,1)'; % venstre kanal i wave-filen udtages

s\_filtered\_voice = filter(b,1,s\_voice\_left); % filtrering med b (FIR)

% \*\*\*\*\* filtreret version \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

s\_filtered\_voice\_norm = s\_filtered\_voice/max(abs(s\_filtered\_voice)); % normalisering

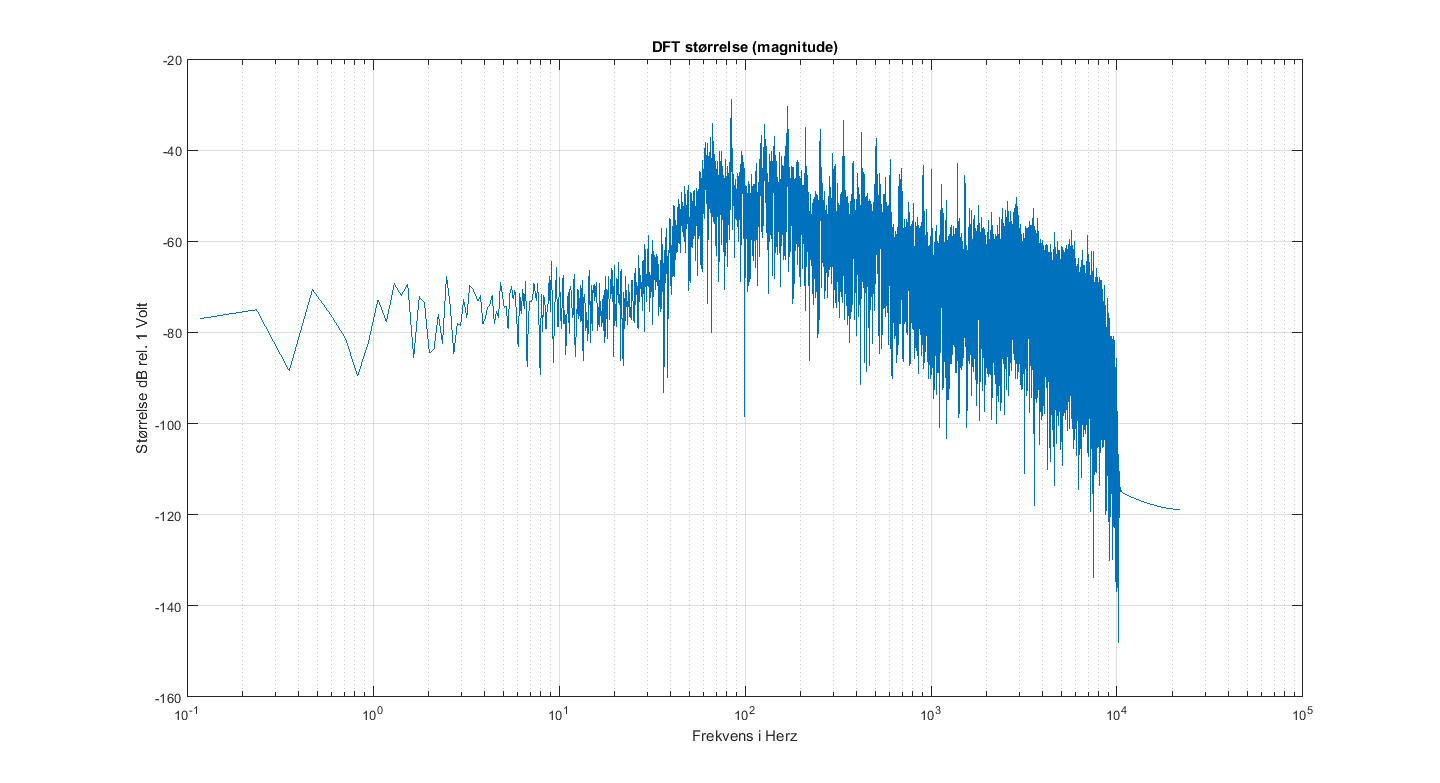
%soundsc(s\_filtered\_voice\_norm, fsample)

end

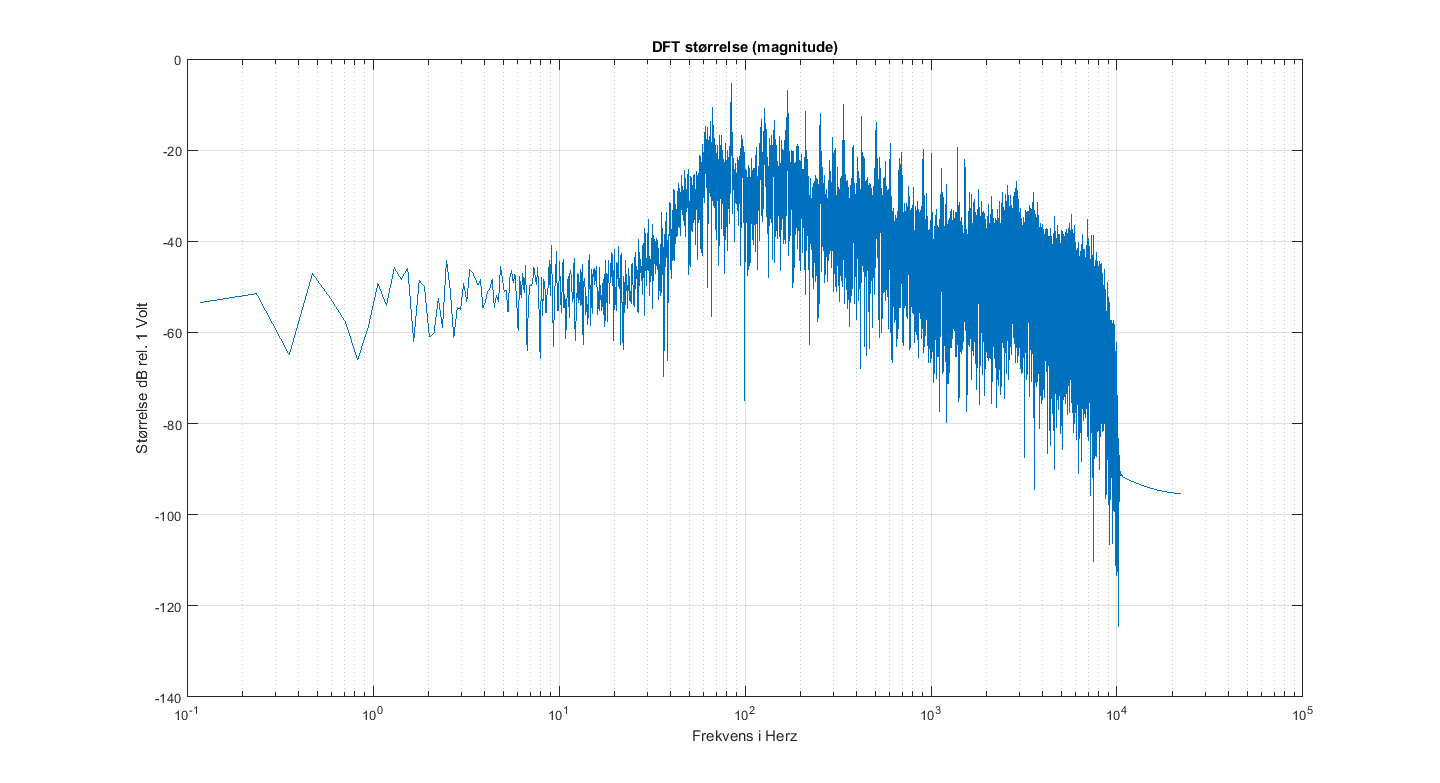
## Volumen

Funktionen volumen, tager et input signal og gange det med et værdi for at forstærke signalet. Dvs. at funktionen skruer op for lyden.

Her er et DFT billede et signal inden det er blevet forstærket.



Og her er et DFT billede efter signalet er forstærket 15 gange. Som det kan ses ligger signalet ved en meget højere dB.



# Equalizer

Equalizeren i vores lydmixer tillader at justere energien i 5 forskellige lige store pasbånd, som er implementeret med FIR filtre, som er fremstillet vha FDA-tool i matlab.

Disse filtre er også påført Hanningvinduer, hvilket gør at filtrenes respons falder hurtigere til ro, men til gengæld bliver filtret en anelse mindre skarpt.

I forbindelse med udviklingen af filtrene er der blevet eksperimenteret lidt med knækfrekvenser og orden. På Figur 13 ses et plot af impulsresponsen for båndpasfiltret 8-12 kHz, med ordenen 1000. På Figur 14 er ordenen fordoblet. Det har den effekt, at filtret bliver mere skarpt, men til gengæld kan man se at delayet før filtrets ønskede virkning indtræffer er fordoblet fra ca 500-1000 samples. Derfor kan man argumentere for, at det måske er fornuftigt at holde ordenen på 1000, da filtrets skarphed egentlig ikke ændres meget markant ved en fordobling af orden, og er derfor ikke det værd.

På Figur 15 er pasbåndets bredde ændret fra 8-12 kHz til 10-12 kHz. Impulsresponsen viser, at filtret bliver mindre skarpt, når man sammenligner med impulsresponsen på Figur 13. Af hensyn til filtrets kvalitet, er der derfor blevet besluttet at holde pasbåndenes bredde på 4 kHz.



Figur 13 – Orden: 1000, knækfrekvenser 8000 og 12000 Hz



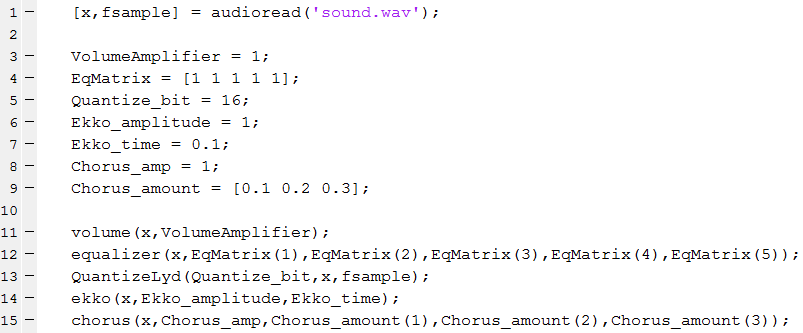
Figur 14 - Orden: 2000, knækfrekvenser 8000 og 12000



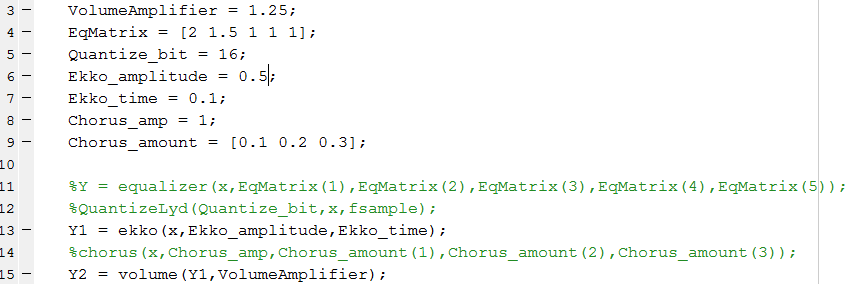
Figur 15 Orden: 1000, knækfrekvenser 10000 og 12000

# Lydmixer implementering og eksempel på brug

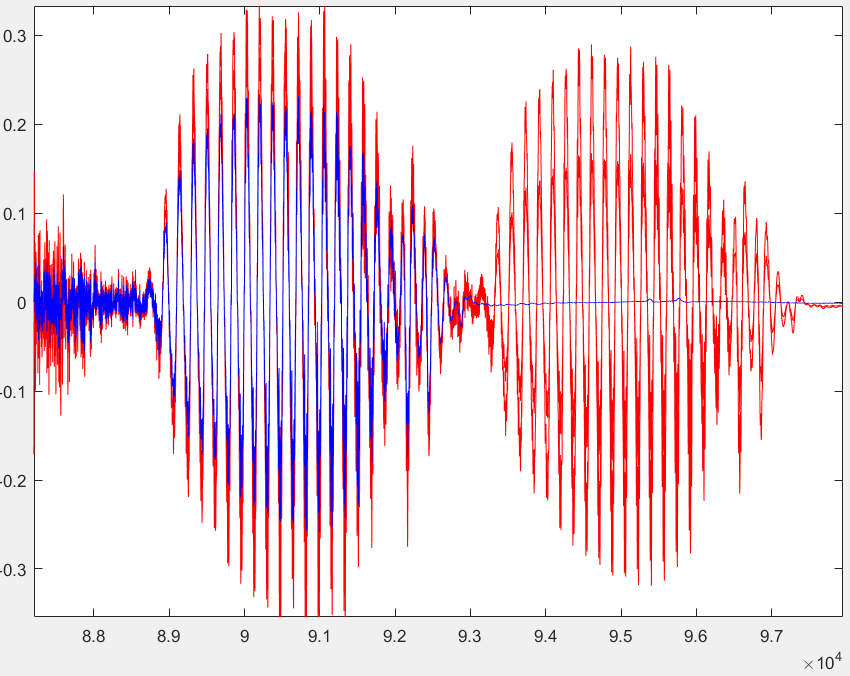
Lydmixerens funktionaliteter er samlet som vist på figur nedenfor



På linje 3-9 er der angivet nogle default værdier til de forskellige lydeffekter. Disse kan ændres som ønsket. På linje 11-15 står de forskellige funktioner skrevet ind. Hvis vi ønsker at øge energien i de lave frekvensspektre, og lave ekko på vores lydsignal, smider vi inputsignalet ind i equalizeren først, og outputtet derfra videre ind i ekko-funktionen.



Her på figur kan det ses at der er påført signalet noget ekko og skruet op for volumen. I linje 6 og 7 er der angivet en ekkotid på 0,1 (hvilket vil sige at der kommer ekko 0,1\*samplefrekvensen forsinket). Amplituden på ekkoet er 0,5 gange det oprindelige. Volumen er ganget med 1,25. På nedenstående billede er der zoomet meget ind på en lille lydimpuls i et klip med en kvinde der taler. På y-aksen er amplitude og på x-aksen er samples. Det blå signal er det oprindelige signal, det røde er det mixede. Det kan ses hvordan amplituden på det mixede signal er højere, at ekkoets amplitude falder, og at ekkoet kommer omkring 4000 samples forsinket, præcis som forventet.



# Diskussion/Konklusion