# 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 1, 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 1 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 2 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 3 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 4 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