DSB

Miniprojekt D

Ekkoeffekt og korrelation

**Gruppe 56**

Alexander Najbjerg Christensen – au482141

Tobias Bach Aunbøl – au328752

Kasper Gnutzmann Andersen – au569735

Indhold

[Formål (Tobias) 3](#_Toc514668874)

[Originalt signal (Alexander) 4](#_Toc514668875)

[Ekko uden filter (Tobias) 4](#_Toc514668876)

[Ekko med fir1-funktion (Kasper) 5](#_Toc514668877)

[Ekko med FIR-filter efter vinduesmetoden (Tobias) 9](#_Toc514668878)

[Beskrivelse af frekvenskarakteristik (Kasper) 12](#_Toc514668879)

[Implementering af fuld ekkoeffekt i matlab (Alexander) 17](#_Toc514668880)

[Eksperimentation med parametre (Kasper) 19](#_Toc514668881)

[Ekkotid bestemt ved autocorrelation (Alexander) 21](#_Toc514668882)

[Konklusion (Tobias) 25](#_Toc514668883)

# Formål (Tobias)

I miniprojekt D har vi pga. interesse valgt at tage udgangspunkt i en case som omhandler ekkoeffekt til elektriske musikinstrumenter. Tidligere blev ekkoeffekt lavet i det analoge domæne ved f.eks. at anvende bånd. I dag er ekkoeffekt blevet mere digitaliseret, enten i form af hardware pedaler eller igennem software på PC. Ekko er pr. definition en gentagelse af lyd med en tidsforsinkelse (delay) og en aftagende volume i forhold til den oprindelig lyd. Formålet med dette projekt bliver derfor at forsøge at implementere og analysere vores egenudviklede, digitale ekko effekt i matlab.

Til udførelsen af miniprojekt D blev der først opstillet nogle brief cases (problemstilling) til at specificere implementering og analyse:

1. Implementer echo effect med FIR filter vha. vinduesmetoden.
2. Foretag en frekvensanalyse af FIR filtret inkl. group delay. Hvordan påvirker effekten lyden set fra frekvensdomænet?
3. Realiser en algoritme i Matlab, som kan lægge ekkoeffekten på et lydstykke.
4. Udvid ekkoeffekten til at kunne håndtere flere ekkoer, eksempelvis 10.
5. Udvid ekkoeffekten til at filtrere ekkosignalet med udviklet FIR filter fra punkt 1.
6. Eksperimenter med parametrene for ekkoet. Hvor lille skal delayet/amplitude være, for at vi ikke længere opfatter gentagelsen som et ekko?
7. Anvend autokorrelation til at finde ekkotider.
8. Konklusion.

Formålet var fra start at projektet efter eksperimenter skulle ende med en digital ekkoeffekt implementeret i matlab som funktion. Funktionen skal have følgende funktionalitet og prototype:

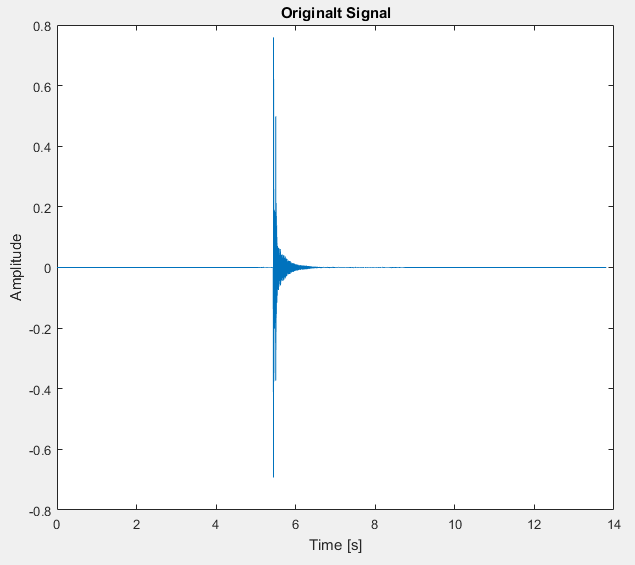
[signal\_echo, Fs] = echoeffect(’lydfil’, echocount, cutoff1, cutoff2, delay)

Ekkoeffekten skal altså tage en lydfil, antal ønskede ekko, tilladt frekvensområde og til sidst hvor lang forsinkelsen skal være imellem ekkoerne.

OBS: Vi har lavet udregninger og skrevet rapporten i fællesskab. Navnene i parenteser er derfor kun indsat pga. krav til opgaven, og er ikke nødvendigvis fyldestgørende for mængden af arbejde pr. person, da vi mener, at arbejdet er gjort ligeligt hele vejen igennem, og teorien forstået tilstrækkeligt for alle personer.

# Originalt signal (Alexander)

Det var ønsket at denne journal skulle omhandle ekko-effekt påført et musikstykke, men det gik hurtigt op for os, at computeren ikke kunne følge med til at arbejde med så store datamængder i vores eget fir-filter. Derfor blev det bestluttet, at der i stedet skulle arbejdes med en kort lydfil, ’glassclinking.mp3’, indeholdende lyden af et glas, der slås på. Den originale lydfil kan ses visualiseret i figur 1.



Figur - Originalt signal 'glassclinking.mp3' plottet med amplitude i forhold til tiden.

# Ekko uden filter (Tobias)

For at få en grundforståelse for implementeringen af ekko, laves først ekko ved at indsætte b-koefficienter, så det passer med at der udsendes ekko med en gg på 0.5 efter et halvt sekund og en gg på 0.2 efter et helt sekund. Det kan ses i koden her:

%Echo effect uden filter:

b\_u\_filter = zeros(1,2\*Fs); %b koefficienter

b\_u\_filter(1) = 1;

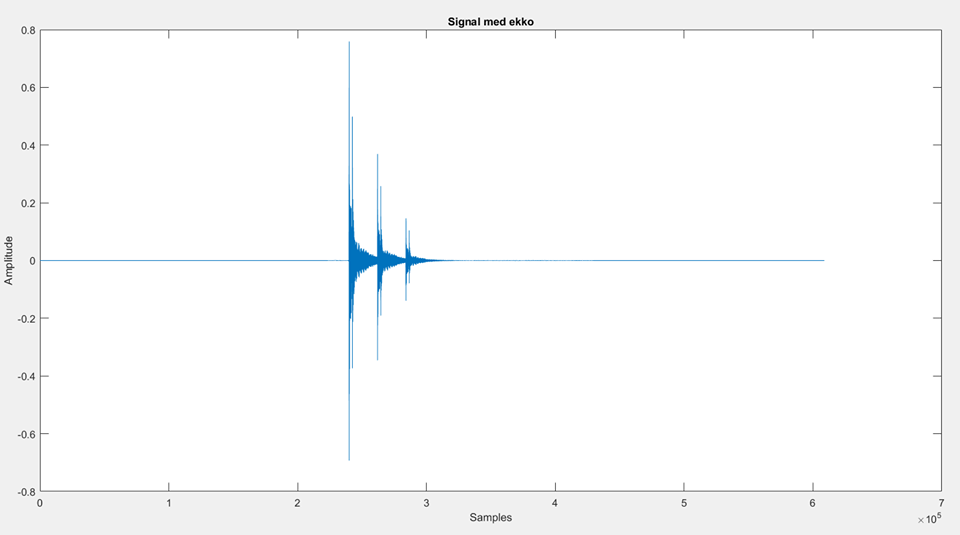
b\_u\_filter(0.5\*Fs) = 0.5;

b\_u\_filter(Fs) = 0.2;

echo\_u\_filter = filter(b\_u\_filter,1,x);

Figur - Kode for ekkoeffekt uden filter

Resultatet af at plotte echo\_u\_filter ses i figur 3.



Figur 3 - echo\_u\_filter plottet i forhold til samples. Bemærk det originale signal efterfulgt af signalet med halveret amplitude efterfult af et endnu mindre signal.

Det ses tydeligt i figur 3, at det originale signal har fået tilføjet ekkoer, og at første ekko er det originale signal med halve amplituder, som det var forventet ud fra matlab-koden. Dertil ses det at andet ekko er betydeligt mindre end det originale signal. Det kan være lidt svært at se, men ekko 2 burde have en amplitude på 20% af det originale signal.

# Ekko med fir1-funktion (Kasper)

Efter at have forstået hvordan b-koefficienterne kan påvirke udgangssignalet, forsøges det at lave ekko med FIR1-funktionen i matlab (figur 4). Når den funktionalitet er på plads, implementeres vores eget filter og FIR1-funktionen erstattes. Mere om det senere i journalen.

Figur - Koden for ekko-effekten ved brug af fir1-funktionen.

%Echo effect med FIR1:

b\_m\_filter = zeros(1,2\*Fs);

b\_m\_filter(1) = 1;

Fn = Fs/2;

b\_m\_filter(0.5\*Fs:0.5\*Fs+50) = fir1(50, [20/Fn 3500/Fn]);

b\_m\_filter(Fs:Fs+50) = fir1(50, [9000/Fn 15000/Fn]);

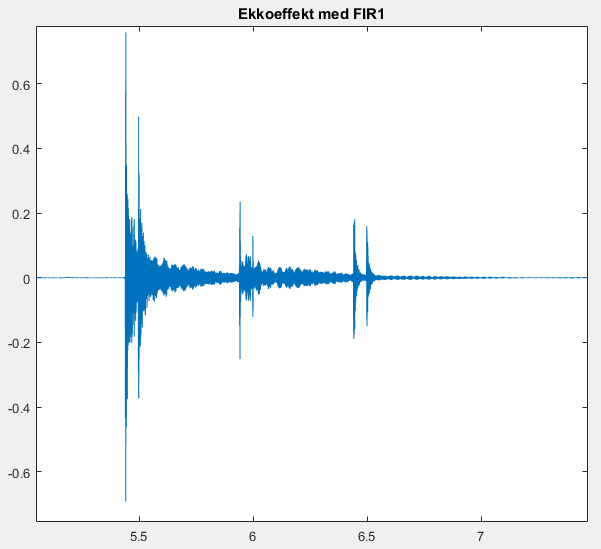
echo\_m\_filter = filter(b\_m\_filter,1,x);

Først laves b-koefficienter (b\_m\_filter) fra 1 til 2\*Fs (fra tiden 0 til 2 sekunder), så vi kan arbejde med tidsforskydning t.o.m. tiden . Herefter tilføjes et enhedsstep på første b-koefficient, så signalet går rent igennem.

Det bestemmes at FIR-filtrene på de to efterfølgende ekkoer skal være 50. orden, hvilket giver 51 koefficienter. Efter et halvt sekund (0.5\*Fs) skal det første ekko komme. Det ønskes at fir1-filteret skal være et båndpas-filter, der lader frekvenser mellem 20Hz og 3500Hz passere. Da der vælges et filter af 50. orden, skal vi lægge de 51 b-koefficienter, der returneres fra fir1-funktionen over i b\_m\_filter fra 0.5\*Fs til 0.5\*Fs+50. På samme måde lægges b-koefficienterne for fir1-funktionen til tiden 1s (Fs:Fs+50). Bemærk at filteret til tiden 1s lader frekvenserne mellem 9000Hz og 15000Hz passere. Det forventes derfor at man vil kunne høre det originale signal efterfulgt af et ekko, der indeholder de lavfrekvente dele af det originale signal efterfulgt af et ekko, der indeholder de højfrekvente dele af det originale signal.

Da b-koefficienterne (b\_m\_filter) nu er opdateret til at indeholde koefficienterne for enhedssteppet og to 50-ordens filtre, kan disse bruges i funktionen filter(), så inputtet kan blive behandlet.

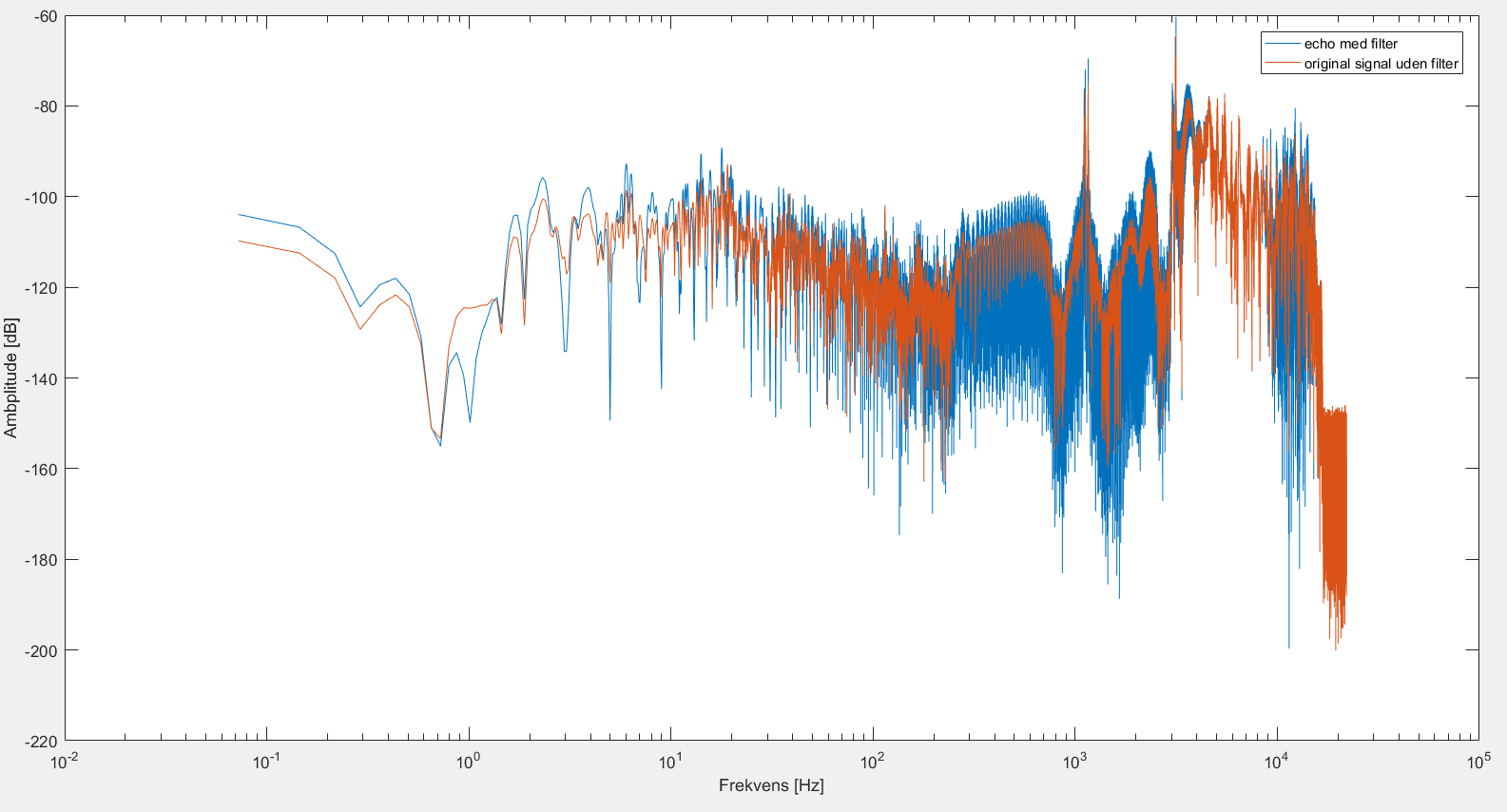
Outputtet af echo\_m\_filter ses i figur 5.



Figur - Ekkoeffekt med FIR1. Bemærk det origniale signal efterfulgt af to ekkoer.

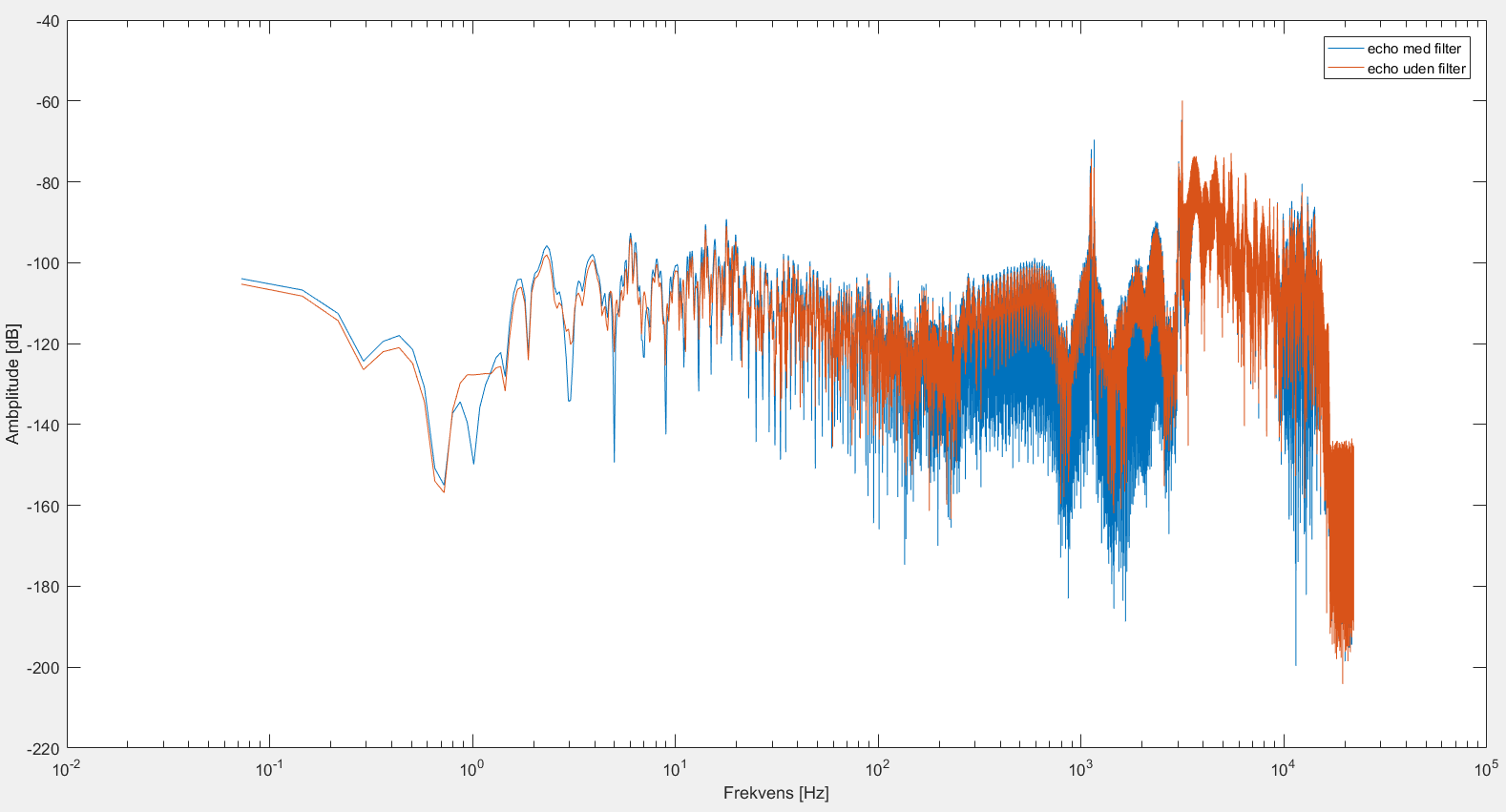
Det kan ses, at der fremkommer to ekkoer på signalet, men at der er ret store, karakteristiske forskelle på de to ekkoer. Lydmæssigt oplevels følgende: Det første ekko ringer længe efter, hvilket skyldes at det er de dybe toner fra glasset, der bliver ved med at ringe i lang tid – Det stemmer godt overens med at lavpasset lader de lave frekvenser gå igennem. Det andet ekko består mest af to store udslag i starten, hvilket skyldes at de høje toner fra slaget på glasset går igennem højpasfiltret. Det kan tydeligt høres at det samlede lydklip med de to ekkoer består af originalen efterfulgt af den lavfrekvente del af originalen efterfulgt af den højfrekvente del af originalen.

En frekvenskarakteristik af lydklippet med de to ekkoer er i figur 6 sammenlignet med det originale signal. Eftersom det første ekko kun består af frekvenser til og med 3500Hz forventes det at effekten ved de lave frekvenser er større end originalsignalet, da de frekvenser er repræsenteret i både originalsignalet og det første ekko. Frekvenser over 9000Hz er både repræsenteret i originalsignalet og andet ekko, hvorfor det må forventes at effekten ved disse frekvenser også er større i ekko-signalet end i originalsignalet. Derimod er frekvenserne mellem 3500Hz og 9000Hz kun repræsenteret i originalsignalet, hvorfor frekvenserne herimellem må være repræsenteret ved samme effekt i både originalsignal og ekkosignal. Dette ses tydeligt i figur 6.



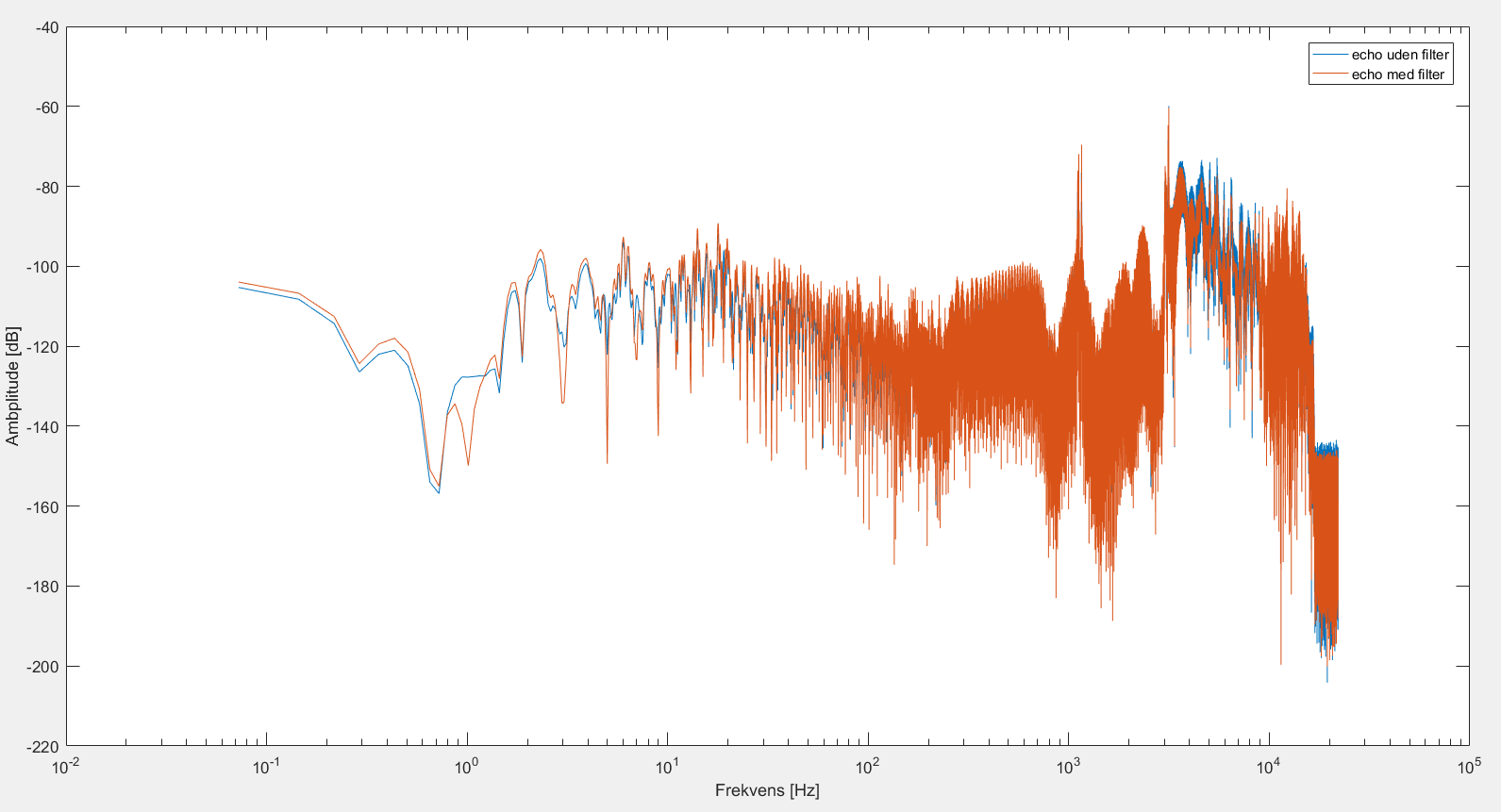
Figur – Frekvenskarakteristik af originalt signal sammenlignet med ekkosignalet. Bemærk ekkosignalets højere effekt mellem 20 Hz og 3500 Hz og mellem 9000 Hz og 15000Hz. Bemærk også at effekten mellem 3500 Hz og 9000 Hz er ens for de to signaler.

For at sammenligne ekkosignalet uden filtrering (fra figur 3) og ekkosignalet med filtrering (fra figur 5), er frekvenskarakteristikken for disse plottet i figur 7 og figur 8. Figur 7 viser at effekten for ekko uden filter er ens igennem hele frekvensspekret. Dette skyldes, at signalet er summen af det originale signal, det originale signal halveret (første ekko) og 0.2 gange det originale signal (andet ekko). Til sammen må det give 1.7 gange effekten af det originale signal. Sammenlignet med figur 7 ses det, at effekten i ekkosignalet uden filter er mere fremtrædende end det originiale signal i figur 6. I modsætning til ekkosignalet uden filter ses det tydeligt at ekkosignalet med filter har større effekt mellem 20Hz og 3500Hz og igen mellem 9000Hz og 15000Hz end ekkosignalet uden filter. Dette skyldes at der i de frekvensspektre går 2 gange det originale signal igennem.



Figur – Frekvenskarakteristik af ekko uden filter i forhold til ekko med filter. Ekkosignalet uden filter er forrest. Bemærk den højere effekt for ekkosignalet med filter mellem 20 Hz og 3500 Hz og igen mellem 9000 Hz og 15000 Hz.

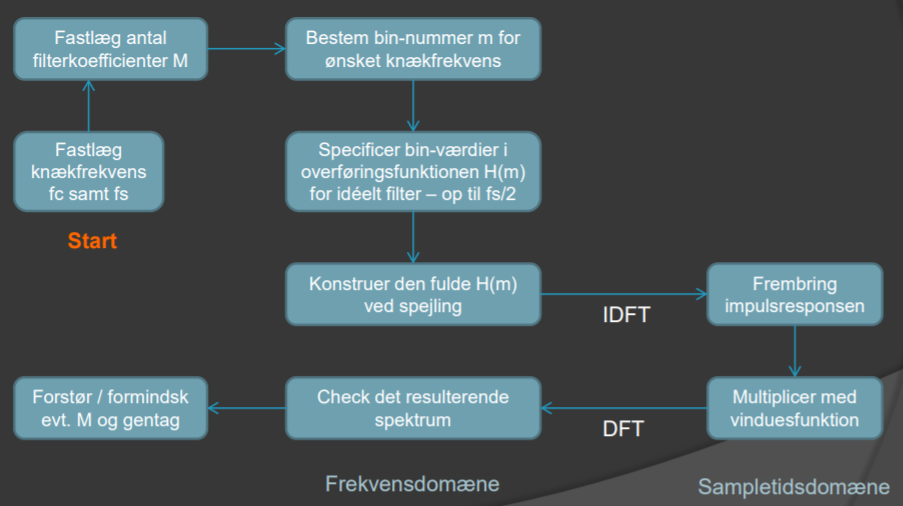
I modsætning til resten af frekvensspektret, er effekten for ekkosignalet uden filter større mellem 3500 Hz og 9000 Hz. Det ses i figur 8. Dette skyldes at der som nævnt er 1.7 gange effekten af originalsignalet i dette spektrum ved ekkosignalet uden filter, hvorimod det for ekkosignalet med filter er originalsignalet, der repræsenterer frekvenser mellem 3500 Hz og 9000 Hz.



Figur - Frekvenskarakteristik af ekko uden filter i forhold til ekko med filter. Ekkosignalet med filter er forrest. Bemærk at der er lavere effekt ved ekkosignalet med filter mellem 3500 Hz og 9000 Hz i forhold til ekkosignalet uden filter.

# Ekko med FIR-filter efter vinduesmetoden (Tobias)

Eftersom implementationen med filter ved brug af FIR1-funktionen, forsøges nu med egen funktion, der konstrueres efter vinduesmetoden. Der tages udgangspunkt i opskriften fra lektion 10 slides som kan ses på figur 9 herunder.



Figur Skabelon fra lektion 10 slides til udvikling af vinduesmetoden

Formålet med at lave filtret til en funktion var så den kunne anvendes i den endelig implementering af ekko effekten, herunder ses implementeringen af vinduesmetoden i matlab

function h\_win = FIRfilterwindow(cutoff1, cutoff2, Fs)

% Fastlæg filterkoefficienter:

M = 500;

freq\_resolution = Fs/M;

freq\_bin1 = cutoff1/freq\_resolution; %%Bestem bin nummer m for ønsket knækfrekvens

freq\_bin\_round1 = round(freq\_bin1);

freq\_bin2 = cutoff2/freq\_resolution;

freq\_bin\_round2 = round(freq\_bin2);

H\_left = [zeros(1,freq\_bin\_round1-1) ones(1,freq\_bin\_round2-freq\_bin\_round1+1) zeros(1,(M/2)-freq\_bin\_round2)];

%^Specificer bin-værdier for overføringsfunktionen op til fs/2

H\_right = fliplr(H\_left(2:end));

H = [H\_left H\_right]; %Konstruer den fulde H(m) ved spejling

h = fftshift(real(ifft(H)));

w\_hanning = hanning(M-1)';

h\_win = h.\*w\_hanning;

end

Design metoden af vinduesfunktionen er derfor bygget op som følge (jf. figur 9):

1. **Fastlæg knækfrekvens fc samt fs**

Samplefrekvensen bliver givet med som parameter til funktionen og fastlægges derfor ikke i funktionen. Igennem dette projekt er der blevet anvendt matlab funktionen audioread, så igennem hele projektet har samplefrekvens lægget på 44100Hz som er samplefrekvens ved sampling af lydfiler.

Knækfrekvens fc bliver også givet med som parameter for at gøre funktionen så versatil som muligt, så den kan anvendes med en hvilken som helst knækfrekvens. Da vi vil anvende funktionen i vores implementering af ekkoeffekten vil det derfor være muligt at give en ønsket knækfrekvens på den endelig ekkoeffekt, med som parameter, derfor bestemmes knækfrekvensen for vinduesmetoden ikke i funktionen.

1. **Fastlæg antal filterkoefficienter, M**

Antallet af filterkoefficienter hænger sammen med filerets orden ved at være en større end filterets orden. Som vi undersøgte i Miniprojekt C, styrer filterets orden, hvor skarpt ”knækket” skal være ved filterets knækfrevenser, altså hvor stejlt den skal skære ved Fc. Vi har valgt at sætte denne til 500, da det vil give en orden på 499 til vores filter. Som det også blev undersøgt i miniprojekt C kræver FIR filtre langt større orden for at være effektive end IIR filtrene, som ikke kan håndtere end særlig stor orden.

1. **Bestem bin-nummer m for ønsket knækfrekvens**

Bin nummer for knæk-frekvensen skal defineres, først bliver frekvensopløsningen fundet, som samplefrekvensen/antal filterkoefficienter. Da bin nummer skal være et hel-tal bliver resultatet derfor rundet til nærmest hel tal vha. matlab funktionen round.

1. **Specificer bin-værdier i overføringsfunktionen H(m) for ideelt filter - op til fs/2 (nyquist-frekvensen)**

Herefter bliver bin værdierne fyldt op med nuller op til fc1-1 og dernæst 1’er fra fc1-1 til fc2+1 til sidst nuller igen. Så der ender med at blive ganget 1’er på i de ønskede frekvensben (mellem fc1 og fc2), og nuller for resten, hvilket svarer til at samples tilhørende frekvensbins uden for fc1 og fc2 bliver ganget med 0.

1. **Konstruer den fulde H(m) ved spejling**

Den fulde H(m) bliver herefter lavet ved at spejle H\_left som blev oprettet jf. punkt 4 vha. matlab funktionen fliplr. H\_left og H\_right bliver til sidst sat sammen.

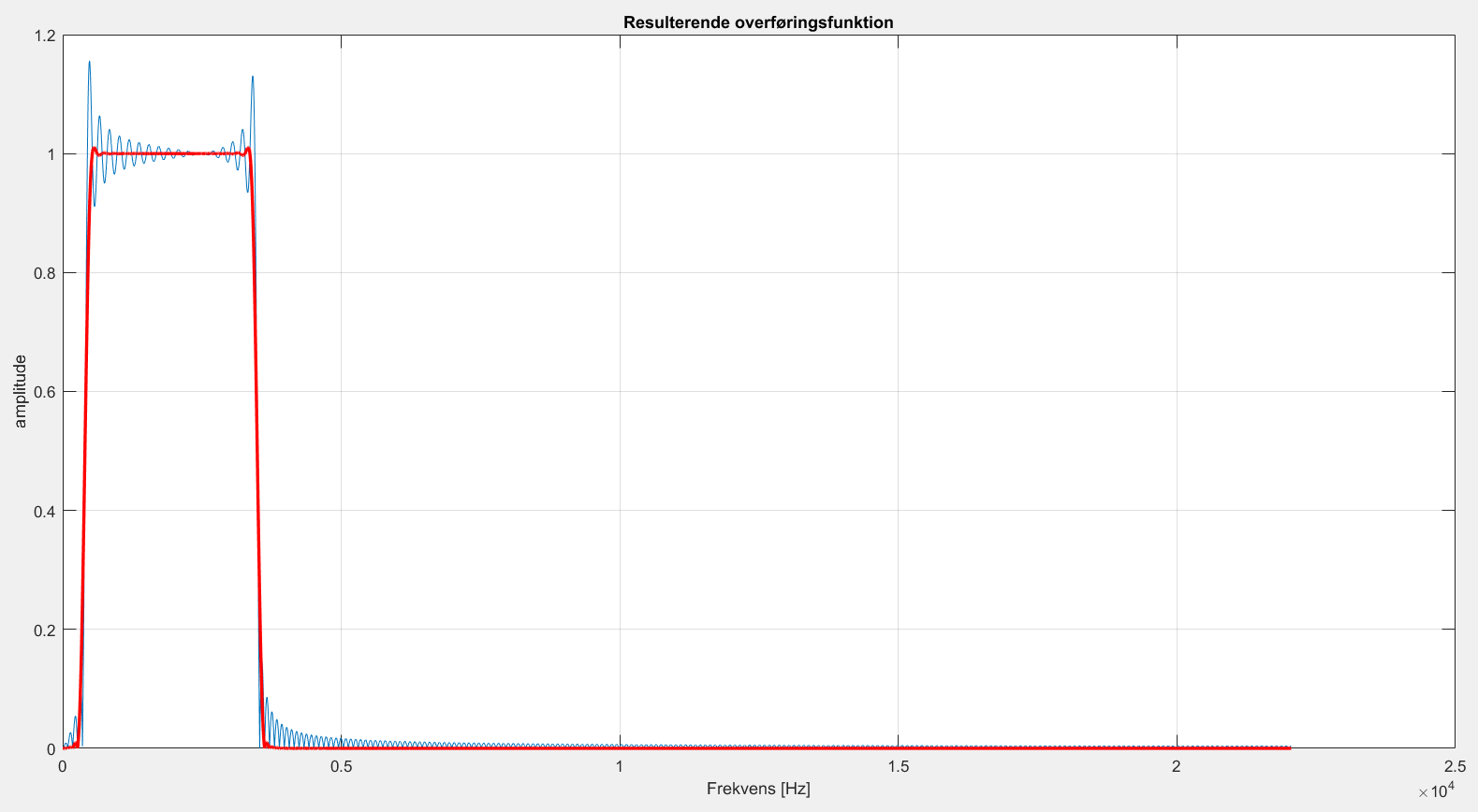
1. **Frembring impulsresponsen**

Impulsresponsen bliver frembragt ved at anvede fftshift. For at kunne multiplicere med hanningvinduet, bliver det lagt over i sampletidsdomænet ved at anvende matlabs inverse fourierttransformation ifft.

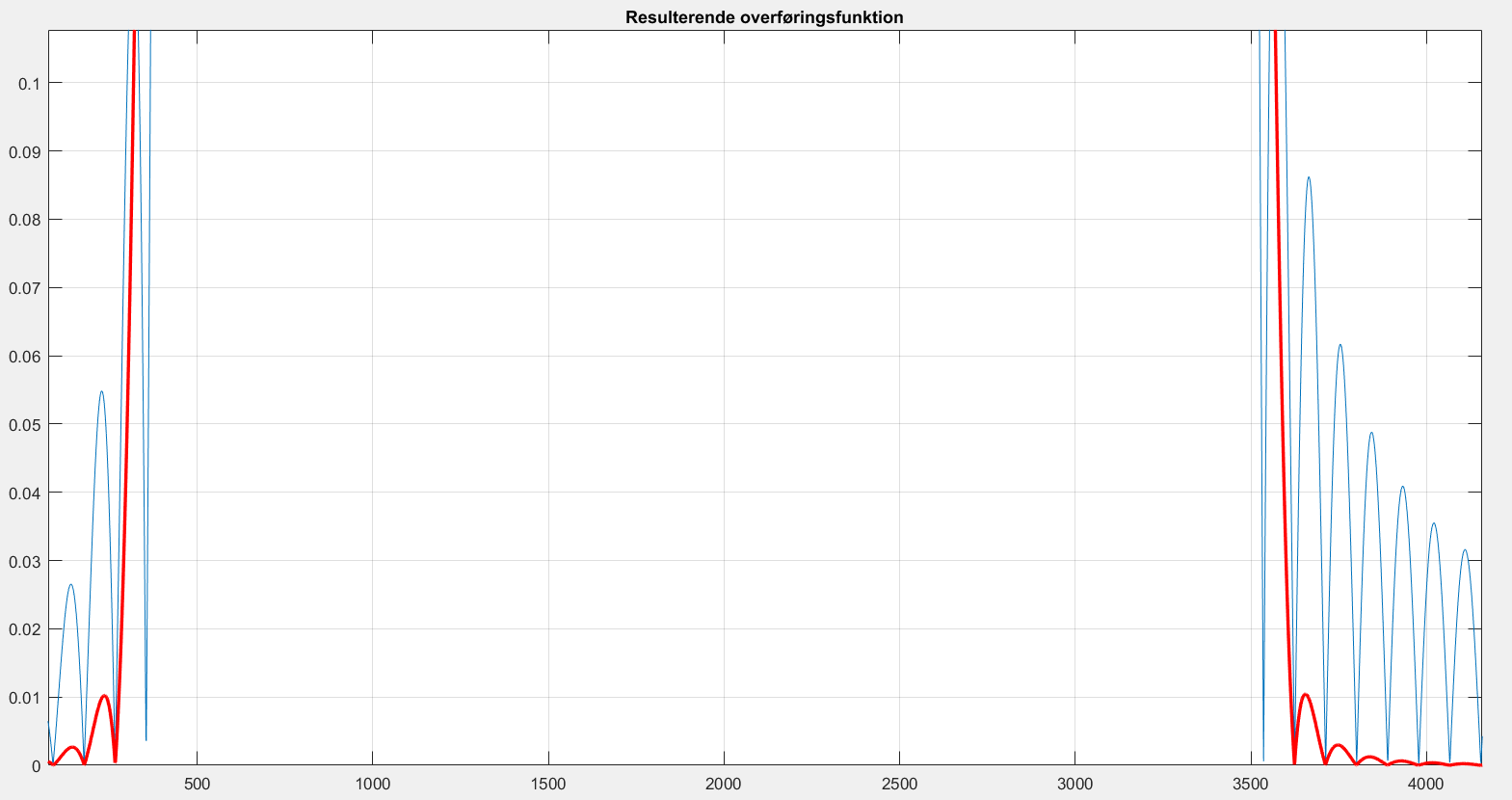
1. **Multiplicer med vinduesfunktionen**

Til sidst bliver et hanningwindow ganget på h over i sampletidsdomænet

Herefter overføringsfunktionen for filteret plottet ved at teste med fc1 = 500, fc2 = 3500, som kan ses på figur 10 herunder og filteret ligger pænt indenfor de to knækfrekvenser som det ses på figur 11.



Figur - Resulterende overføringsfunktion for egetudviklet FIR-filter efter vinduesmetoden. I dette tilfælde ses et båndpasfilter med gain på 1 mellem 500 Hz og 3500 Hz.



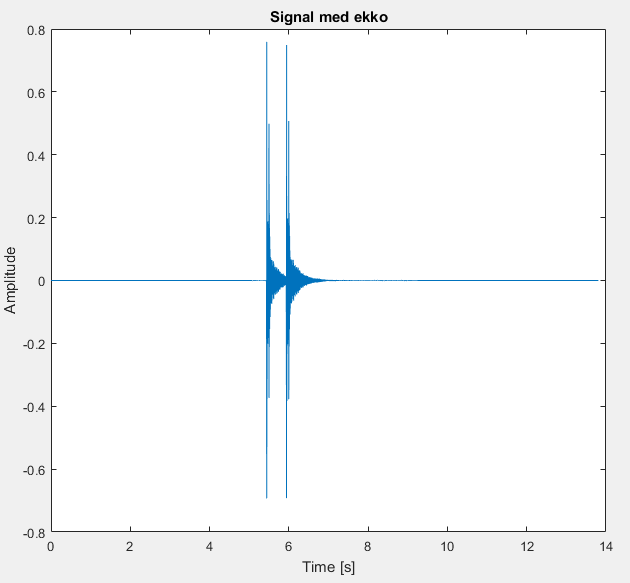
Figur - Resulterende overføringsfunktion for egetudviklet FIR-filter efter vinduesmetoden. I dette tilfælde ses et båndpasfilter med gain på 1 mellem 500 Hz og 3500 Hz.

Da udviklingen af FIR filteret ikke blev prioriteret højere i dette projekt, blev der ikke eksperimenteret yderligere med filterkoefficienter. Flere filterkoefficienter ville have fået den resulterende overføringsfunktion til at lægge tætter på de to knækfrekvenser, men til proof-of-concept for vores endelig implementering, beslutede vi os for at 500 var et fint antal af filter koefficienter.

## Beskrivelse af frekvenskarakteristik (Kasper)

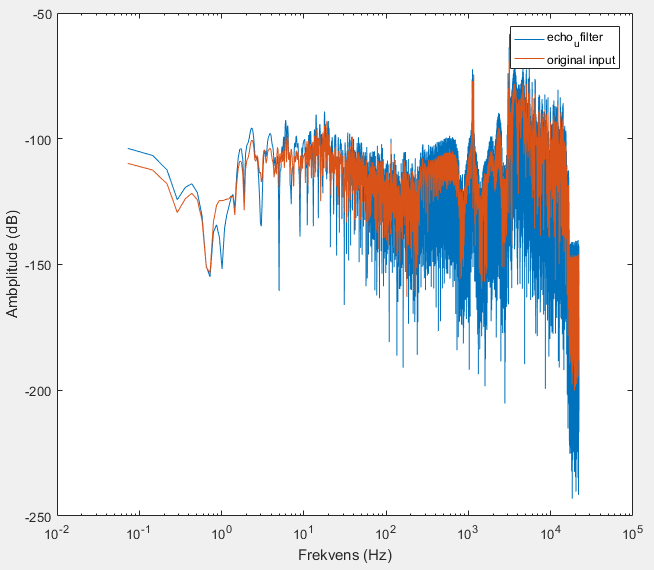
Ligesom ekko frembragt ved brug af FIR1-funktionen, har vi lavet frekvenskarakteristik for ekko frembragt ved FIR-filter opstillet efter vinduesmetoden.

På figur 12 ses et samplingssignal repræsenterende det originale signal samt enkelt ekko uden filter.



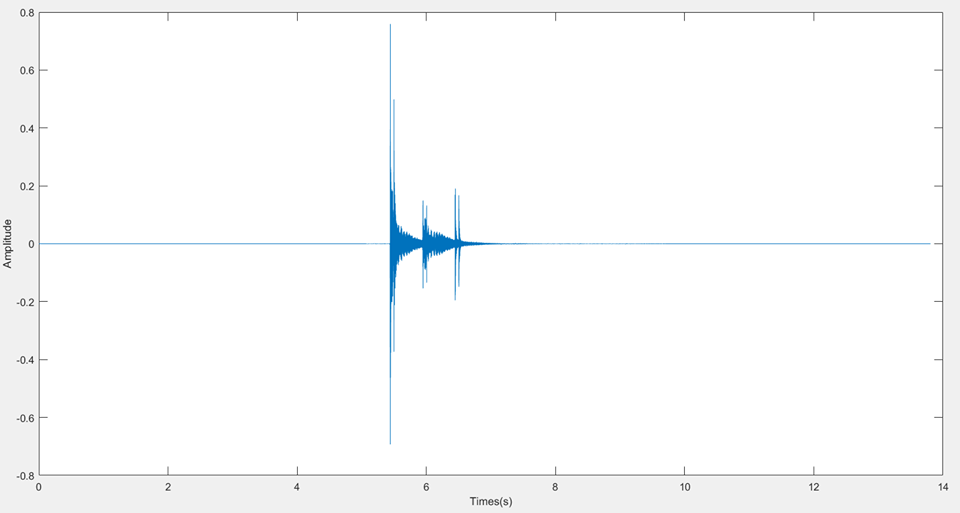
Figur 12 – Signal med ekko efter 0.5 sekunder. Original-signalet og ekkoet er ens.

På figur 13 ses frekvenskarakteristikken for det originale signal (orange), som er første instans af signalet set på figur 12. Desuden kan frekvenskarakteristikken for hele signalet med ekko (uden filter) også ses på figuren (blå). Sammenlignes de to karakteristikker, er det tydeligt, at der er større energitæthed i signalet med ekko. Dette stemmer godt overens med forventningen, da det oprindelige signal optræder to gange, når vi har et ekko.



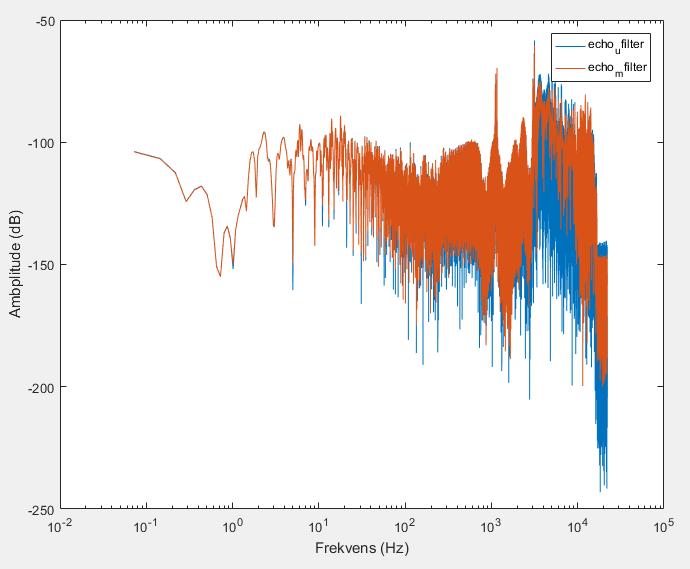
Figur 13 – Frekvenskarakteristik af originalt input (orange) og signal med ekko uden filtrering (blå).

På figur 14 ses samplingssignalet repræsenterende ekkosignalet med filter (vinduesmetode). I dette signal ses to ekkoer; et hvor filteret er opstillet som lavfrekvenspasbånd med 0,5 sekunds delay, samt et hvor filteret er opstillet som højfrekvenspasbånd.



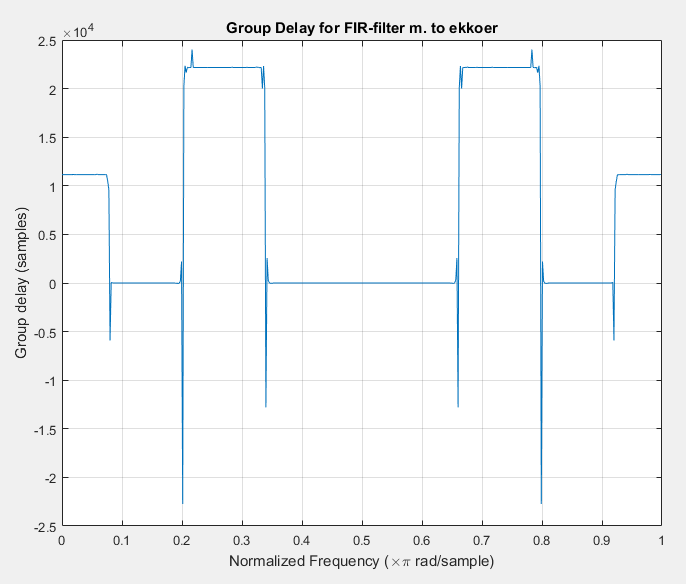
Figur 14 – Signal med to ekkoer. Første ekko indeholder kun frekvenser mellem 100 Hz og 2 kHz. Andet ekko indeholder kun frekvenser mellem 6 kHz og 20 kHz.

Betragtes nu figur 15, ses frekvenskarakteristikkerne for ekkosignaler frembragt uden filter (blå) og frembragt ved brug af filteret opstillet efter vinduesmetoden (orange), set på figur 14.  
Ekkosignalet uden filter er identisk til det, som ses på figur 13. Sammenlignes karakteristikkerne for de to ekkosignaler, ses de som nær identiske for lave og høje frekvenser. Der er et frekvensbånd herimellem, hvor karakteristikken ligner det, for det originale signal. Dette kan forklares ved at filteret i dette eksempel er opbygget således en ekkoeffekt optræder med 0,5 sekund delay for et frekvensbånd aflæst til 100Hz-2kHz, og med et delay på 1 sekund for et frekvensbånd aflæst til 6kHz-20kHz. Frekvensbåndet mellem 2kHz og 6kHz mangler altså i begge ekkosignaler, hvorfor energitætheden for echo\_m\_filter fremstår som for det originale signal i dette område.



Figur 15 - Frekvenskarakteristik af ekkosignal med filtrering (orange) og ekkosignal uden filtrering (blå).

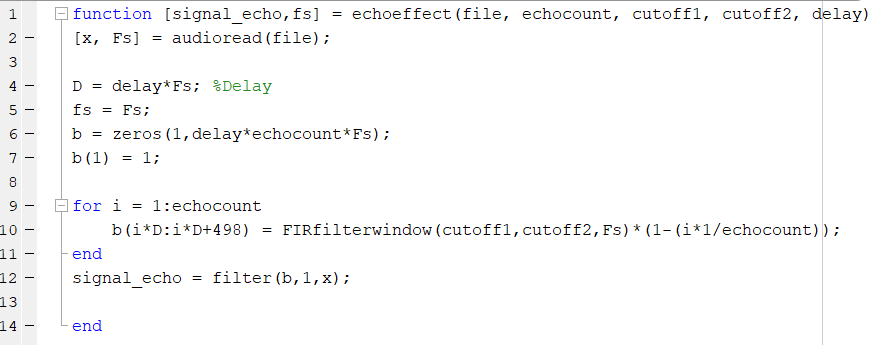
På figur 16 ses repræsentation af group delay for signalet set på figur 14. Group delay beskriver graden af distortion af signalet, og det ønskes oftest ensartet over alle frekvenser, hvorved distortion er mindst.  
Men for dette signal er ekkoerne jo netop en distortion af signalet, som ønskes. Vi ønsker først at forsinke nogle lave frekvenser med 0,5 sekund, og derefter at forsinke nogle høje frekvenser med 1 sekund. Dette er netop, hvad figuren illustrerer.   
Det første delay aflæses til samples, og det andet delay til samples. Disse svarer netop til 0,5x og 1x gange nyquistfrekvensen, hvorved de to delays svarer til 0,5 og 1 sekund.



Figur 16 – Group Delay plot for ekkosignalet fra figur 14.

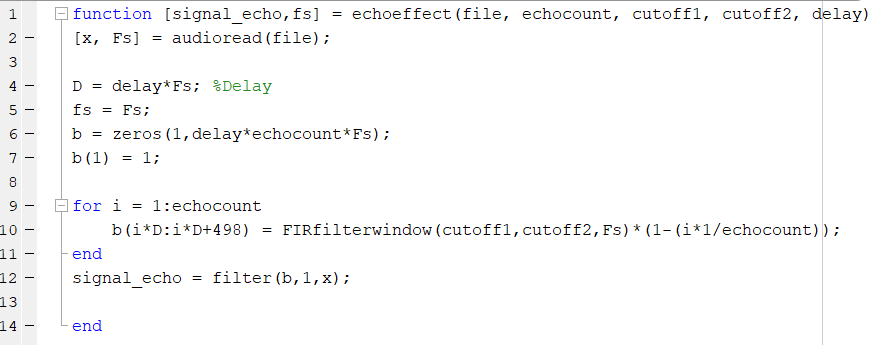
# Implementering af fuld ekkoeffekt i matlab (Alexander)

Efter de foregående eksperimenter blev de hægtet sammen for at implementere en fuld ekkoeffekt i matlab, som skulle have prototypen vist i figur 17.



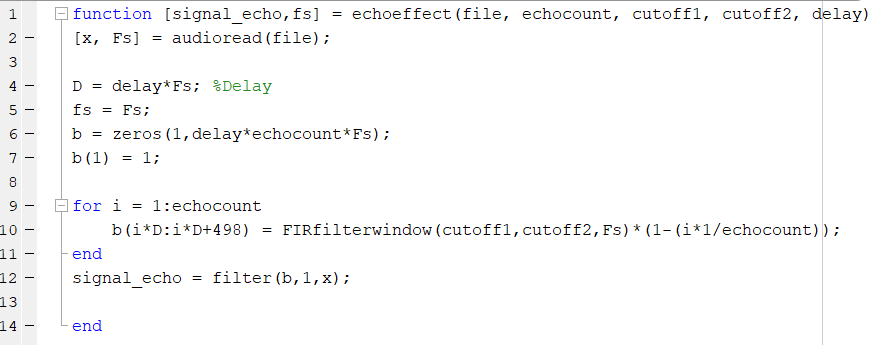
Figur - prototype til ekkoeffektfunktion

Implementeringen af funktionen ecchoeffect i matlab kan ses herunder på figur 18-21.



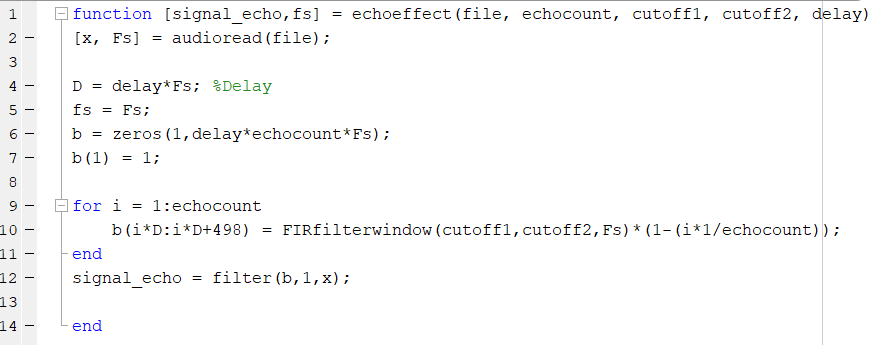
Figur - Implementering af ecchoeffect - del 1

Som det ses udfra figur 18, bliver audioread funktionen først anvendt til at læse filen, som ekko effekten skal lægges på. Herefter bliver delay udregnet i samples ved at gange delay parameteren (i sekunder) med sampling frekvensen.



Figur - Implementering af ecchoeffect - del 2

B-koefficienterne bliver lagt i en vektor (figur 19), der er på størrelse med antallet af ønskede ekkoer ganget med forsinkelsen i samples. Alle B-koefficienter sættes først til 0, hvorefter første b-koefficient sættes til 1. Dette medfører at originalsignalet går rent igennem til tiden 0.



Figur 20 - Implementering af ecchoeffect - del 3

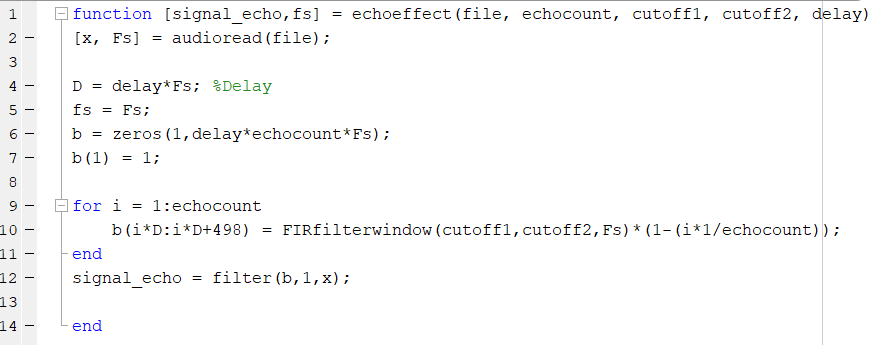
For hvert ekko (i), der skal lægges på signalet, bliver FIRfilterwindow-funktionen kaldt (figur 20), og filterkoefficienterne, der bliver returneret fra funktionen, bliver lagt ind på b-koefficienterne til tidsforsinkelsen D\*i. Eftersom der i FIRfilterwindow bliver returneret 499 b-koefficienter, lægges de over til og med i\*D+498.

For at opnå en ekkoeffekt med faldende effekt, ganges alle returnerede b-koefficienter med en brøk:

Denne vil sørge for at der vil laves ekkoer, der falder i intensitet. Det er efterfølgende gået op for os, at ovenstående metode vil give os et sidste ekko med en amplitude på 0, hvilket giver et resultat med et ekko for lidt i forhold til parameteren echocount, som burde være det ønskede antal ekkoer. Dette kan dog hurtigt ændres ved at lægge en til nævneren:

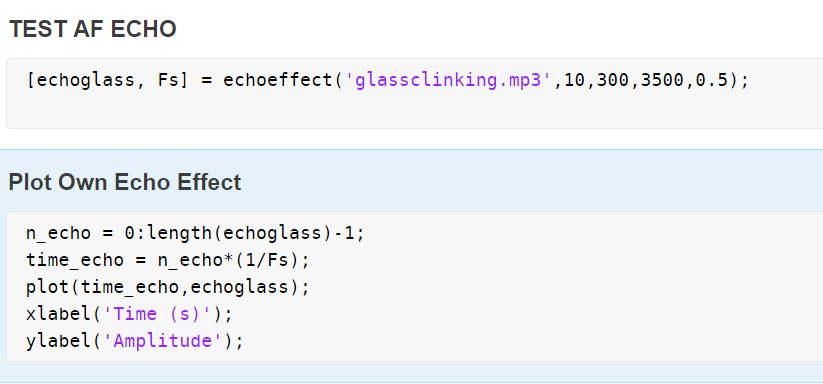
Men eftersom journalen er skrevet efter data er indsamlet, fortsættes der med et manglende ekko.

Til slut i funktionen ecchoeffect kaldes funktionen filter() for at filtrere originalsignalet med de nye b-koefficienter, som det kan ses i figur 21.



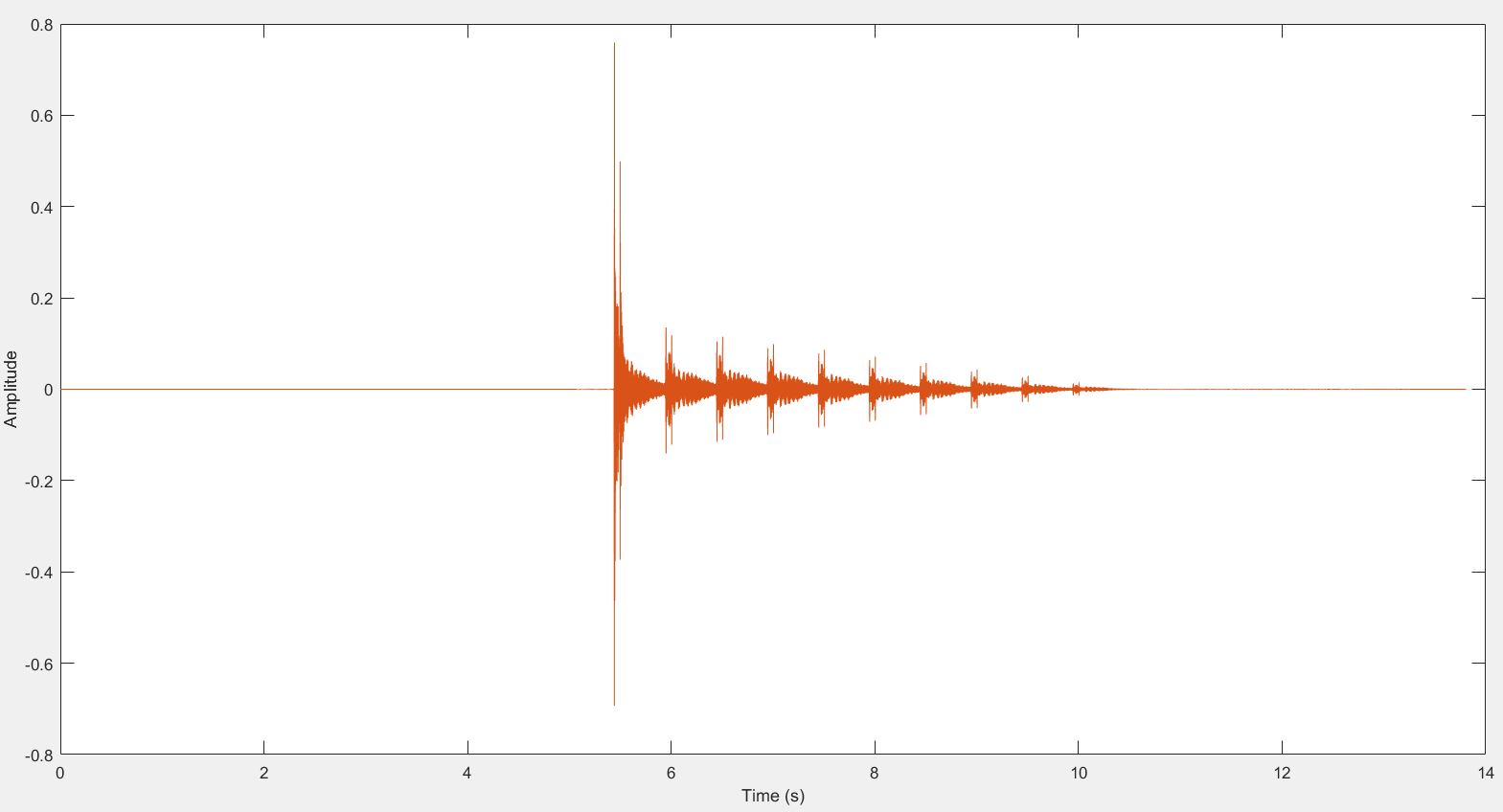
Figur - Implemenring af ecchoeffect - del 4

I figur 22 er ecchoeffect-funktionen kaldet med ønsket om ti ekkoer med et halvt sekunds forsinkelse, der lader frekvenser mellem 300Hz og 3500Hz passere hvorefter retur-værdierne er plottet.



Figur 22 - Test af echoeffect-funktionen

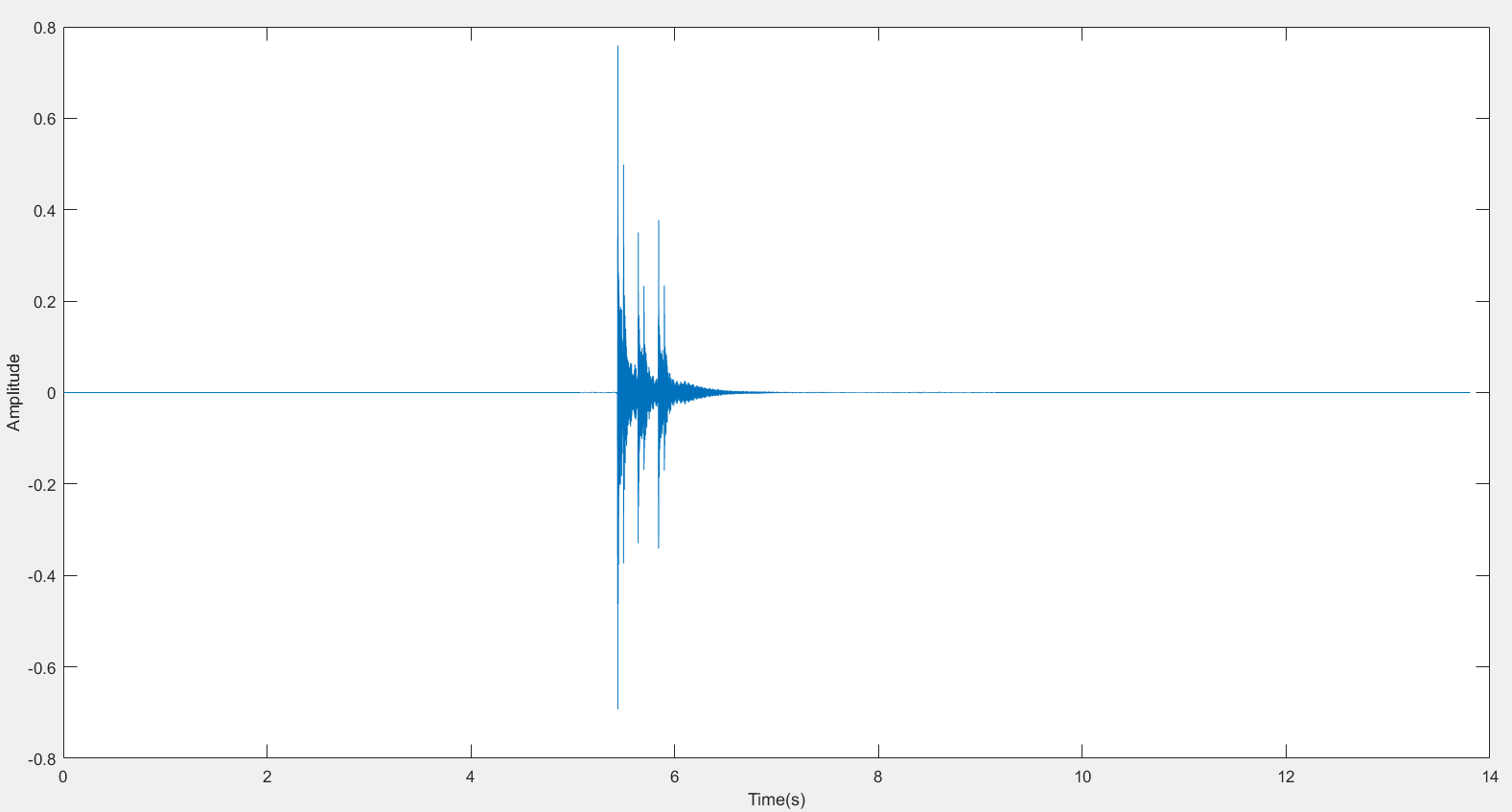
Resultatet af ovenstående test giver 23, hvor det kan ses at det originale signal sendes igennem til tiden 0 efterfulgt af 9(!) ekkoer med et halvt sekunds forsinkelse mellem hver.



Figur - Resultat af implementering af echoeffect-funktionen. Bemærk det originale signal efterfulgt af 9 ekkoer med aftagende effekt, der kommer med et halvt sekunds forsinkelse.

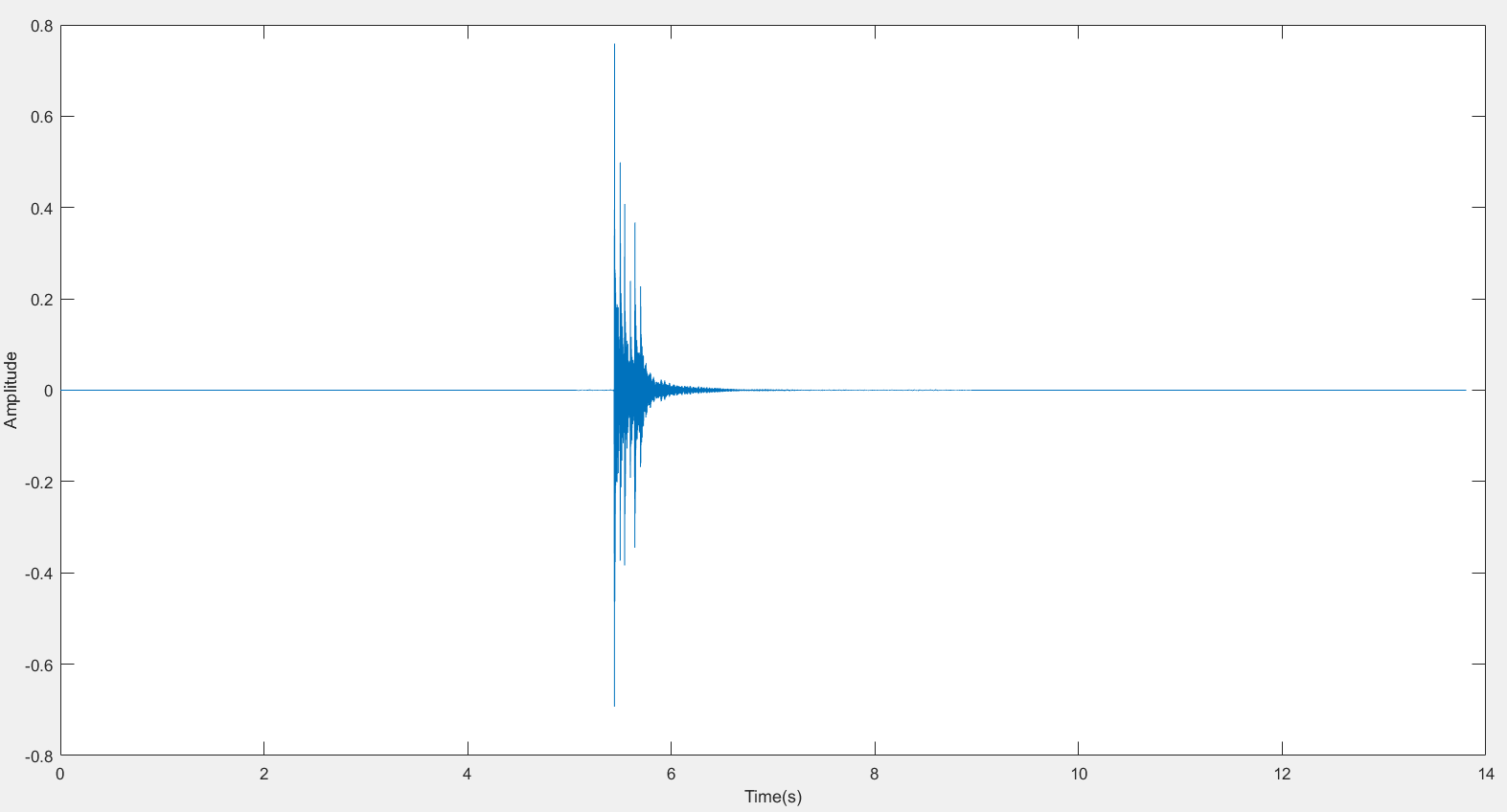
# Eksperimentation med parametre (Kasper)

Til at starte med undersøges det hvor kort delay ekkoet skal have for ikke at lyde som ekko længere. For at skrue på så få parametre som muligt, sendes originalsignalet igennem efterfulgt af to ekkoer med den halve amplitude. Først undersøges det om 0.2 sekunders delay er nok (figur 24).



Figur - Originalt signal efterfulgt af to ekkoer med 0.2 sekunders delay.

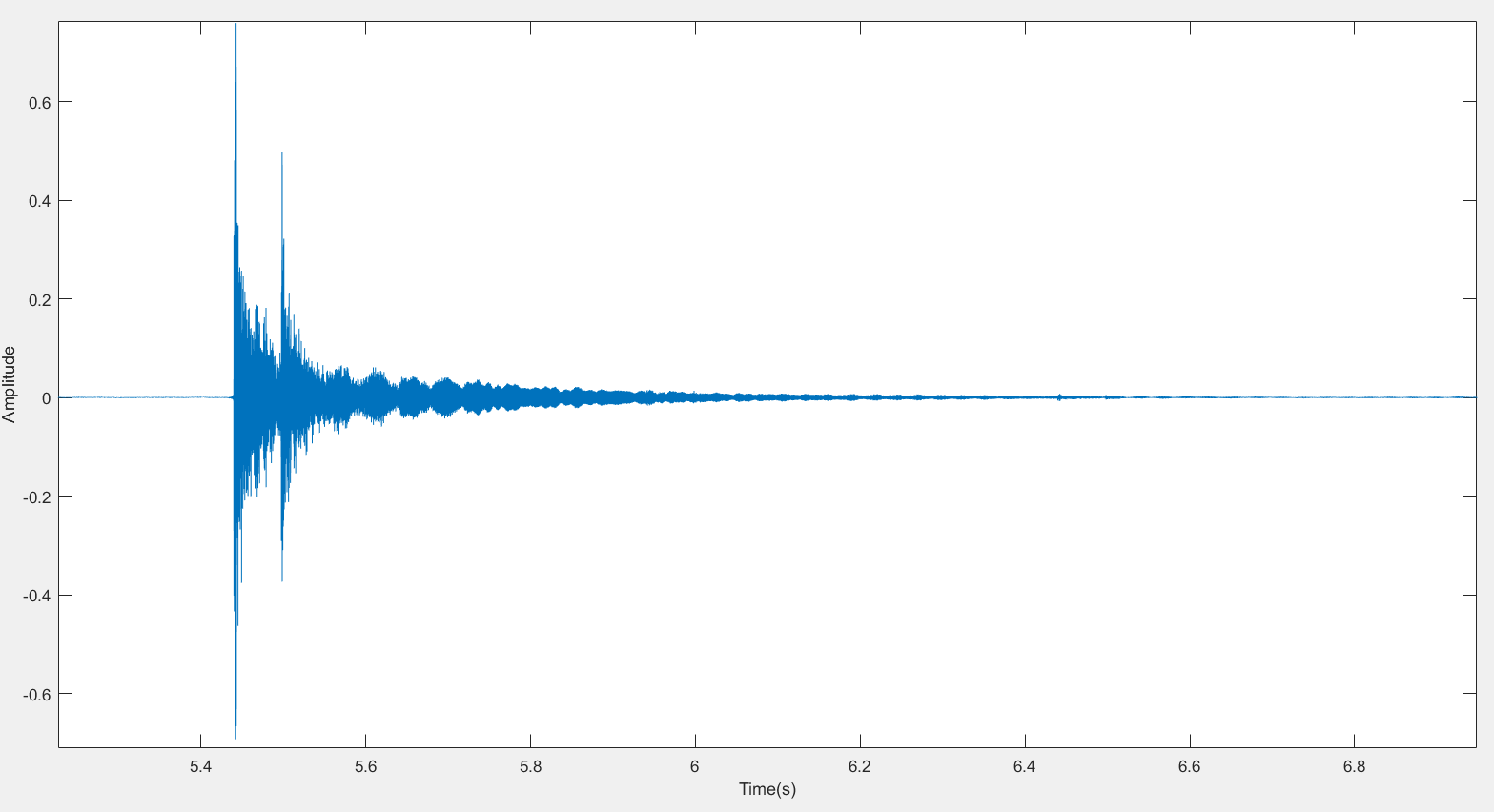
Ekkosignalet i figur 24 kan sagtens adskilles lydmæssigt, og lyder stadig som tre seperate lyde. Derfor nedsættes delay til 0.1 sekund (figur 25).



Figur - Originalt signal efterfulgt af to ekkoer med 0.1 sekunders delay.

Som det kan ses på Figur 25 er de tre signaler så tæt sammen, at de ikke længere kan registreres som tre forskellige lyde. I dette tilfælde, hvor der arbejdes med en lydeffekt, registreres ekkoet derfor indtil signalerne overlapper hinanden. Det vil måske ikke være tilfældet ved sang, hvor den såkaldte chorus-effekt måske vil opfattes tidligere. Der er muligvis en sammenhæng mellem den meget korte lydfil og opfattelsen af et ekko som værende et egentligt ekko, men eftersom vores filter bruger for lang tid på at arbejde på længere musikstykker, kan denne teori ikke undersøges til bunds i denne journal.

For at undersøge hvor længe et ekko lyder som et ekko ved lavere amplituder, sættes delay til 0.5 sekunder, hvor det er bevist at ekkoet lyder som et faktisk ekko. Dertil ændres amplituden på de to ekkoer til 0.01 for at høre, om det stadig lyder som ekko (figur 26).



Figur - Originalt signal efterfulgt af to ekkoer med 0.5 sekunders delay og amplituder på 0.01.

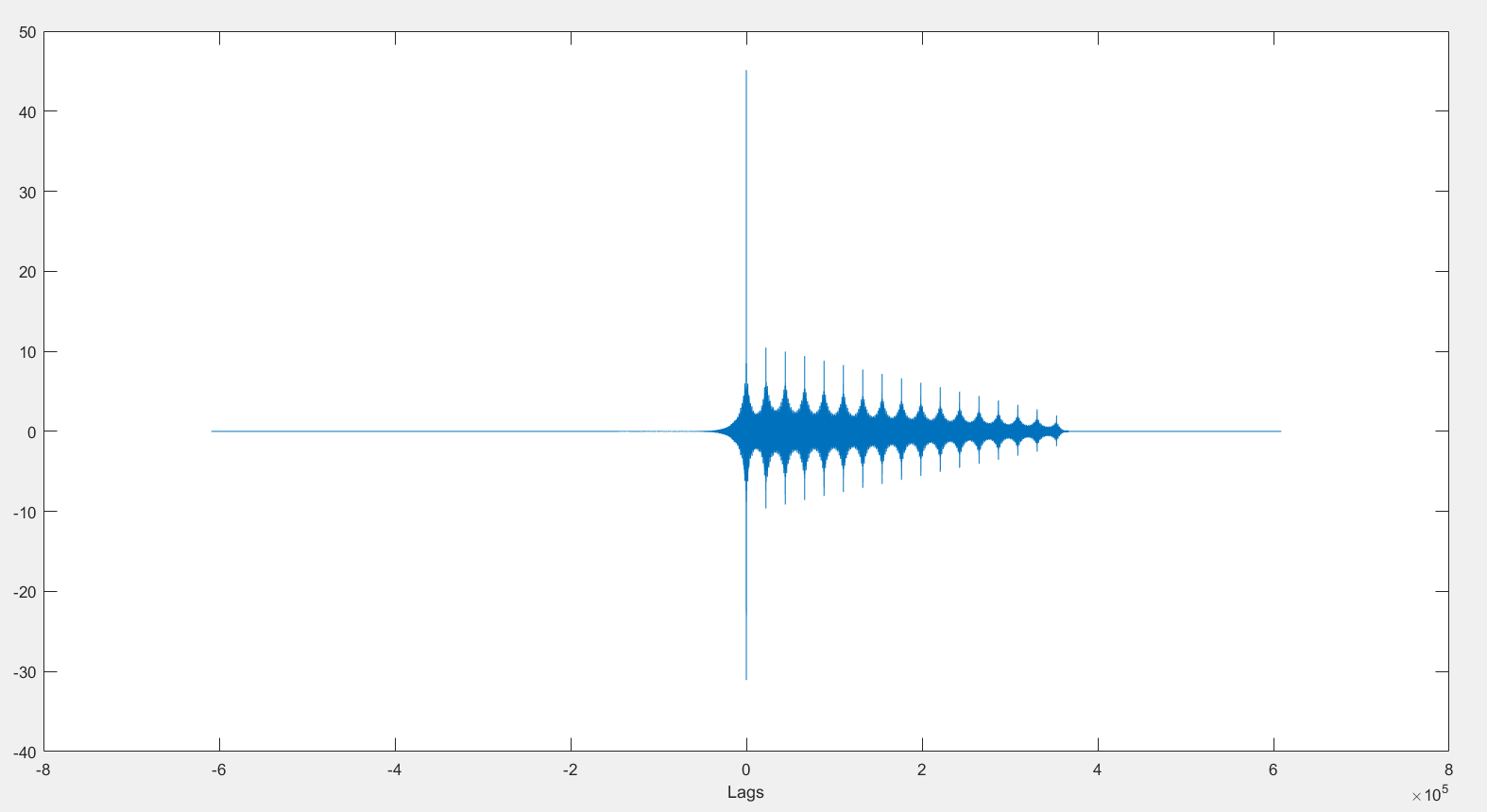
Selvom det i figur 26 ikke rigtig kan ses, at der er ekko på signalet, kan det stadig tydeligt høres. Det må derfor konkluderes, at et ekko kan høres så længe der rent faktisk kommer lyd ud af højtaleren, og at ekko-effekten kun kan ændres ved at gøre delayet mindre.

# Ekkotid bestemt ved autocorrelation (Alexander)

Cross correlation bruges til at finde ensartethed mellem et kendt signal og ukendte signaler. Det udregnes ved

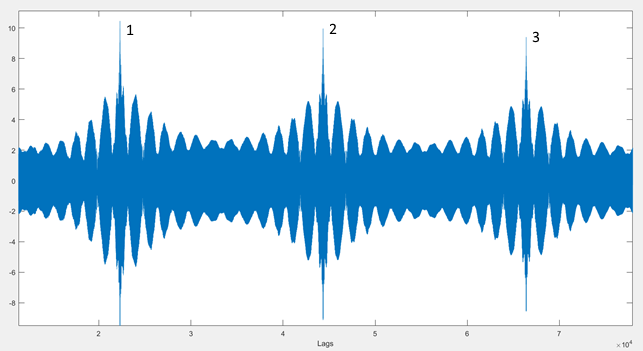
Hvor er det komplekse kojugat af og er tidsforskydningen også kaldet *lag*. I ord betyder det, at hver diskret værdi, , af det originale signal bliver ganget med en diskret værdi fra det ukendte signal, . Da der er tale om diskrete værdier, summeres multiplikationerne for at finde integralet.

I matlab bruges funktionen xcorr(), der kan returnere cross correlation-værdierne og lag-værdierne. Plottes disse, fås en graf som set i figur 27.



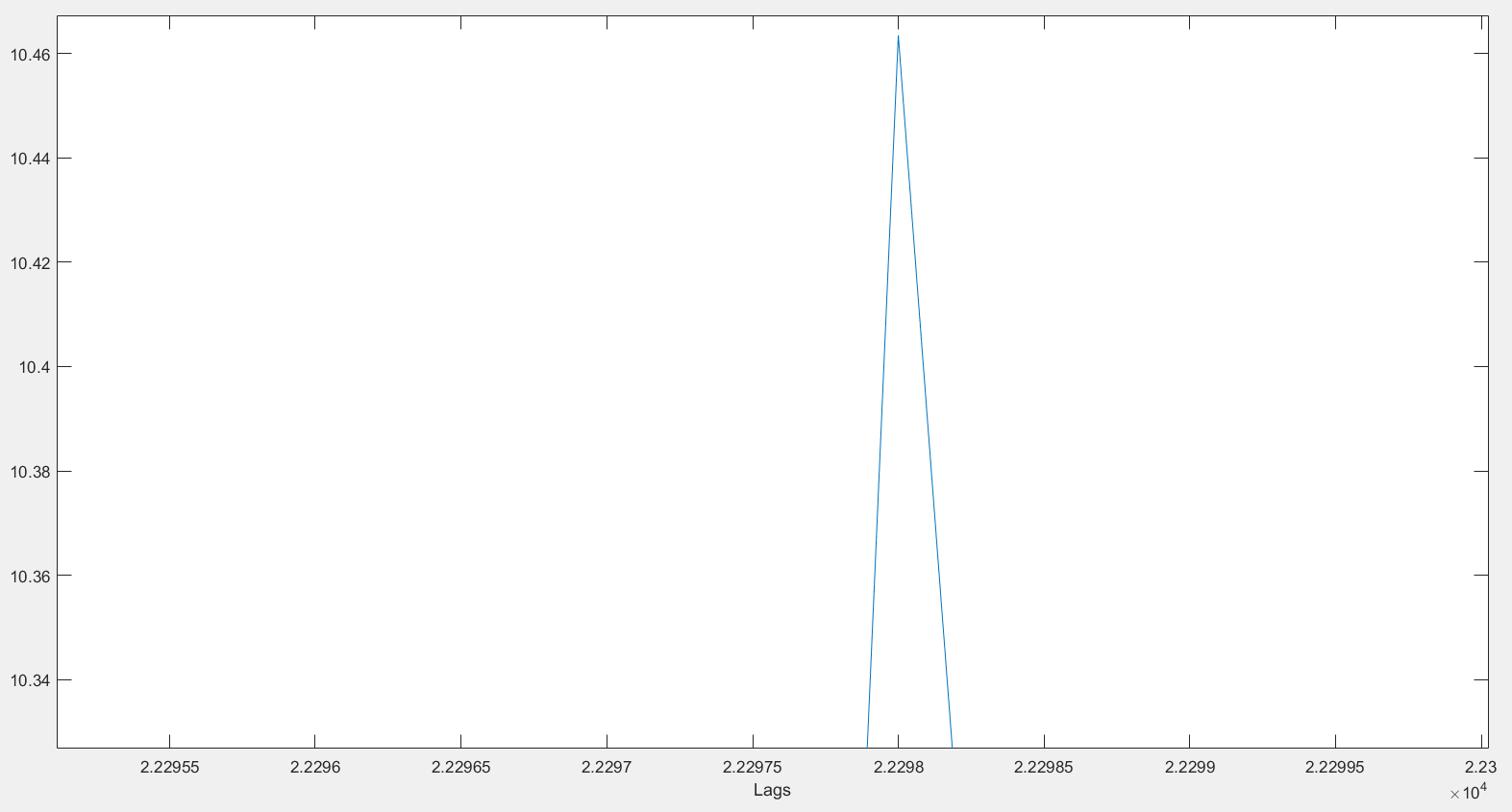
Figur 27 – Cross correlation mellem det originale signal og ekkosignalet plottet i forhold til lags.

Først og fremmest bør det bemærkes at der er stor sammenligning mellem figur 23 og figur 27. Det ses tydeligt at korrelationen er stor mellem det originale signal og ekko-signalet til tiden 0. Dette skyldes at ekkosignalet starter med det originale signal, hvilket naturligvis vil forårsage en meget stor korrelation, da signalerne er ens. Efterfølgende ses peaks, der stemmer rigtig fint overens med ekkoerne i ekko-signalet. Ved at finde lag-værdierne der hvor korrelationen er højest, kan man finde tidsforsinkelsen på ekkoerne. Dette skyldes at forskydes én sample af gangen, og forskydningen derfor kan findes i tid ved at gange lag-værdien med sampletiden. På figur 28 ses de tre første ekko-signaler tydeligere.



Figur - Viser de tre første ekkoers korrelation med det originale signal.

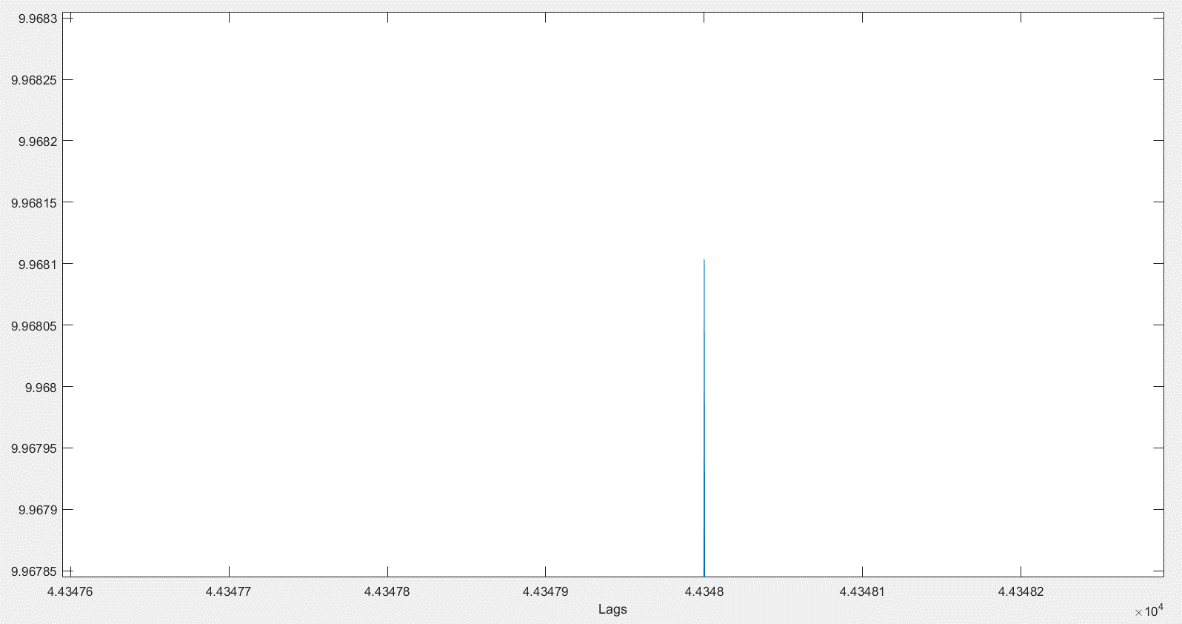
Ved at zoome ind på figur 28 kan man finde lag-værdien for de peaks, der ses. På figur 29 ses det første peak ved sample .



Figur - Cross Corelation plottet i forhold til lags. Zoom in på første ekko på ekkosignalet. Det ses at højeste korrelation for ekko 1 findes ved lag 222980.

For at beregne tiden for ekkoet ganges med sampletiden:

Eftersom det er ønsket at vores ekkofunktion skal tilføje ekkoer hvert halve sekund, må det kunne erfares at cross correlation fint kan bruges til at finde originale signaler i ukendte signaler. For at sikre, at det ikke bare er et tilfælde, at den første peak i cross correlation ligger efter et halvt sekund, undersøges også ekko 2. Peak for ekko 2 ses i figur 30.



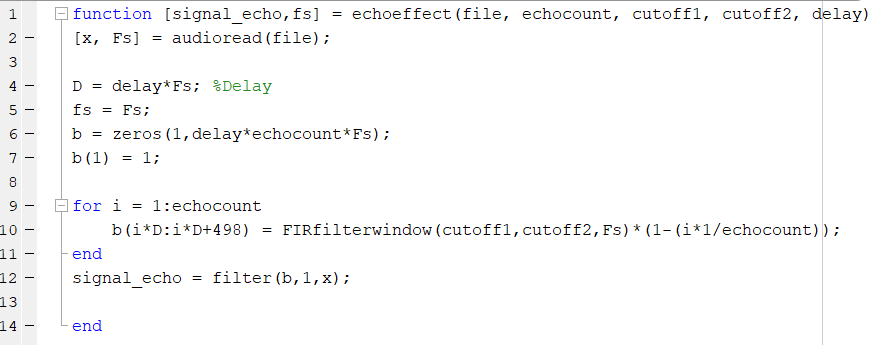
Figur - Cross Corelation plottet i forhold til lags. Zoom in på andet ekko på ekkosignalet. Det ses at højeste korrelation for ekko 2 findes ved lag 443480.

Det er tydeligt at se i figur 30, at højeste korrelationsværdi for ekko 2 er ved lag 443480, som svarer til sample 443480. Tidsforsinkelsen findes ved:

Eftersom ekko 1 kommer med et halvt sekunds forsinkelse og ekko 2 kommer med et helt sekunds forsinkelse samtidig med at resten af korrelations-peaks ligger med samme afstand til hinanden, må det konkluderes at vi med xcorr-funktionen kan bruge cross correlation til at finde kendte signaler i ukendte signaler.

# Konklusion (Tobias)

Efter indledende tests med indbyggede funktioner i matlab er det lykkedes at konstruere en funktion, der tager en lydfil, et antal ekkoer, to knækfrekvenser (der danner et båndpas) samt et delay, og returnerer det resulterende signal og samplefrekvensen, som vist nedenfor:



Igennem frekvensanalyse er det blevet vist, at funktionen echoeffect virker efter hensigten, og at effekten indsætter echocount antal ekkoer med de tilladte frekvenser inkluderet.

Ved hjælp af funktionen er det undersøgt, hvilke parametre, der afgør hvorvidt resultatet lyder som et ekko eller ej. Det blev her konkluderet at et ekko ikke mister sin effekt ved at sænke lyden på ekkoerne. Derimod er det yderst vigtigt at ekkoerne ikke har et for kort delay, hvilket vil medføre en chorus-effekt i stedet. I tilfældet med glas-lydfilen indtræffer chorus-effekt ved delays under 0.1 sekunder. Større delays medfører tydelige ekkoer.

Til sidst er det undersøgt, om man ved hjælp af autokorrelation og det originale signal kan finde tiden for ekko-signaler. Efter at have lavet et signal med ni ekkoer er der brugt cross correlation til at undersøge, hvornår signalet ligner det originale signal mest. Ved visuel analyse af cross correlation mellem det originale signal og ekko-signalet er det konkluderet at cross correlation kan bruges til at finde kendte signaler i et ukendt signal, og at ekko-tiden derved kan findes.