Svingninger og Bølger

Oblig1

Matthew Terje Aadne

Oppgave 1)

Ved å ha den initielle frekvensen f0 = 3000, og en ny nærliggende frekvens f1, fant jeg at ejg kunne høre forskjell mellom dem når f1=3002. Så differansen df= 2. Derfor er en god estimering av Q-verdien Q=f0/df = 3000/2 = 1500.

Program:

```
%Samplingfrekvensen
Fs = 22050;

N = 2^12;

T = N/Fs;


A = 0.8;
f1 = 3000;
f2 = 3002;
df = f2 - f1

x  = 0:(1/(Fs)):1;
% Her lager jeg de to distinkte signalene
func1 = A*sin(2*pi*f1*x);
func2 = A*sin(2*pi*f2*x);

% Her setter jeg dem sammen til et kontinuerlig
signal
func = [func1, func2];
sound(func,Fs)

Q = f1/df
```

Oppgave 2)

Her er programmet:

```
frekvens = 4000

%samplingfrekvens
Fs= 22050



t1 = 0:1/Fs:0.012; %Kort tidsinterval
t2 = 0:1/Fs:0.5;  %Lengre tidsinterval
wavplay(sin(t1*2*pi*frekvens),Fs);
pause(2)

wavplay(sin(t2*2*pi*frekvens),Fs);
```

Her fant gjennom gjentatte forsøk at det minste tidsintervallet som kreves for å identifisere en tone er på 0.012s.

<u>Sammenlikning mellom oppgave 1 og 2:</u>

Vi vet at antall perioder kraften må virke før man oppnår en tilnærmet stabil svingetilstand, øker når Q-verien øker. Derfor vet vi at siden Q-verien øker med frekvensen (Man kan høre forskjell i en frekvensdifferanse på 2 for frekvenser mellom 100 og 10000) vil det ta flere perioder å høre en høy frekvent lyd enn en lav frekvent lyd. Men siden perioden er avhengig av frekvensen, P =1/f, vil perioden bli mindre når frekvensen øker. Jeg har gjennom forsøk avgjort at det er lettere å oppfatte høyere frekvenser over et kort tidsinterval, enn lav frekvente lyder over et kort tidsinterval. anta at h(f)=P(Q(f)) er en funskjon som angir antall perioder før svingningen blir stabil. La g(h(f)) være en funksjon som angir hvor lang tid det tar før vi oppfatter en tone. Siden g angivelig blir mindre når f øker, og g blir større når h(f) blir større, må f vokse fortere enn h(f), så h'(f)<1. Dermed vil en økning iantall perioder bli oversteget av økningen i frekvensen.

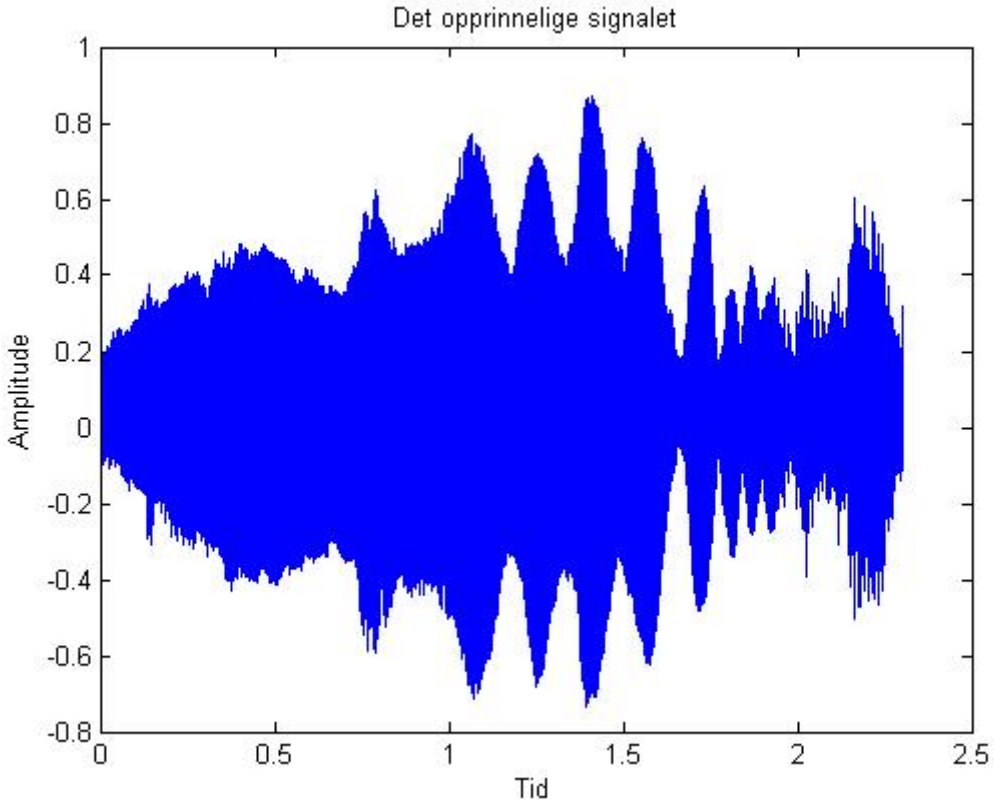
Oppgave 3)

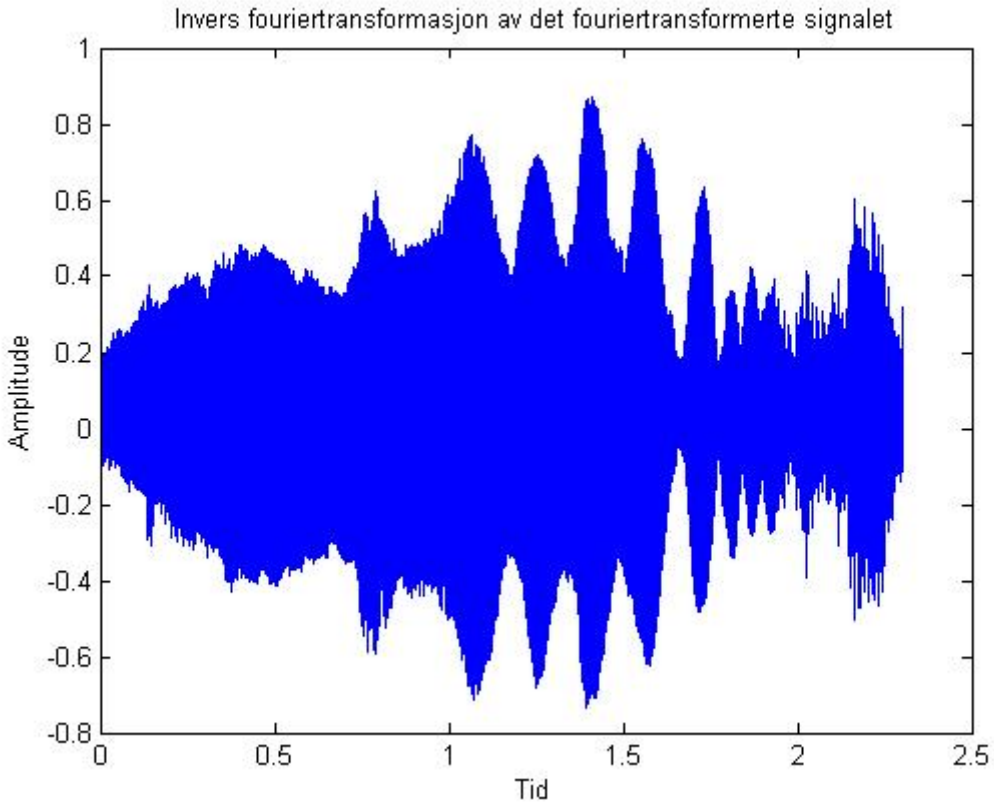
Her er programmet: for YellowBostich2 trenger man bare å bytte fra lydfil2 til YellowBostich2 i wavred i andre linje av programmet.

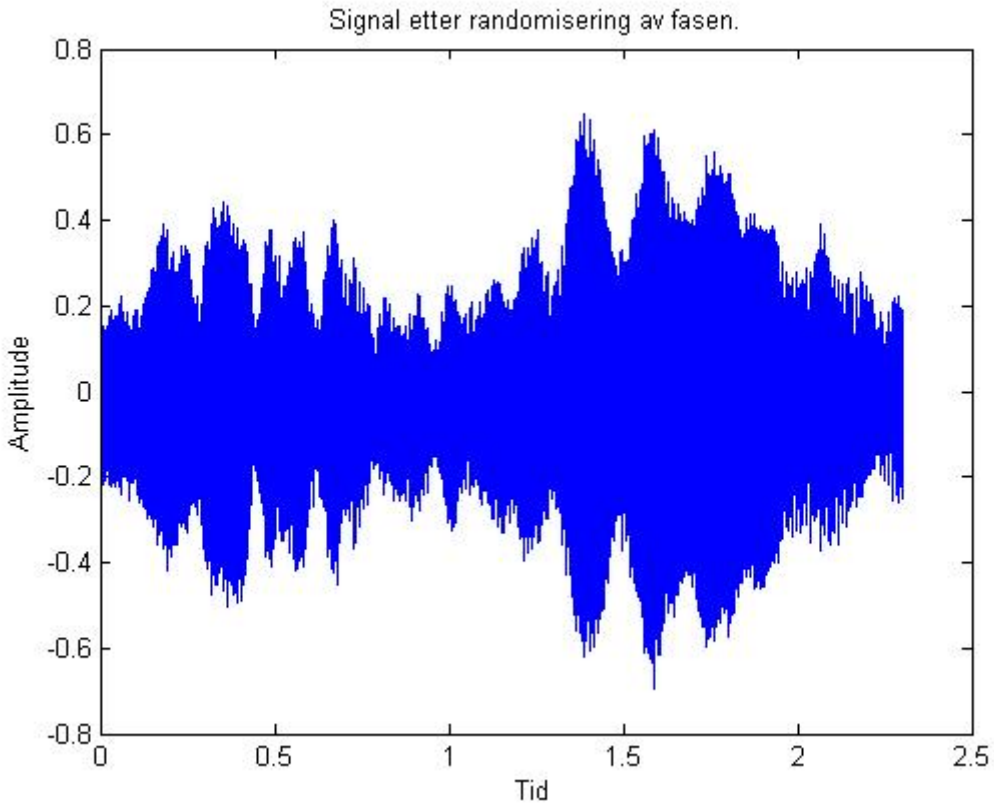
```
format long
E = wavread('lydfil2.wav');
Fs = 44100;
N = length(E);
T = 1/Fs;
%tidsvektor
t = (0:(N-1))*T;

% opprinnelige lydsignalet
figure(1);
plot(t,E)
title('Det opprinnelige signalet')
xlabel('Tid')
ylabel('Amplitude')

% Fourier transform av lydsignalet
F = fft(E,N)/N;
%invers fourier av fouriertransformasjonen
invF = ifft(F,N)*N;
figure(2)
% Her plotter jeg det nye signalet.
plot(t,invF);
xlabel('Tid')
ylabel('Amplitude')
title('Invers fouriertransformasjon av det
fouriertransformerte signalet')


%her spiller jeg den inverstransformerte av fourier
transformasjonen
wavplay(invF,Fs);

pause(3)

% Her lager jeg to arrays med nuller. Jeg Legger den
randomiserte fasen
%i arrayen R, og nullstiller fasen i array H. For å
randomisere fasen, tar
%jeg hvert element fra det fouriertransformerte signalet,
finner absouluttverdien, og lager et nytt komplekst tall
%der absoluttverdien er bevart. Dette gjør jeg ved å la
vinkelen theta være
%et vilkårlig tall fra 0 til 2pi, for så å la reel delen av
det nye
```

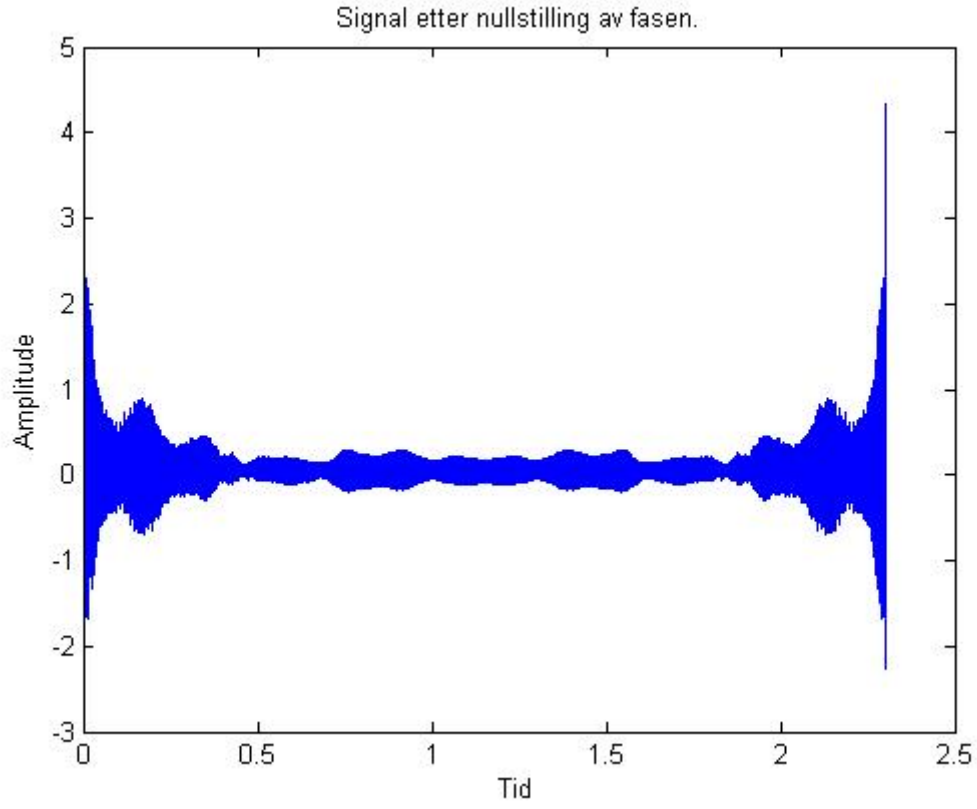
```
%komplekse tallet være absoluttverdien av elementet ganget med
cos(theta),
%og den nye imaginærdelen absoluttverdien av elementet ganget
med
%sin(theta). For å nullstille fasen bevarte jeg igjen
absoulutt verdien av
%de komplekse tallene i fouriertransformasjonen av signalet,
og jeg endret
%elementene til kun å ha en reel-del slik at reel-delen til
hvert nye element
%ble absoluttverdien til det gamle elementet, og imaginærdelen
ble satt til null.
R = zeros(length(F),1);
H = zeros(length(F),1);
for x=1:length(F)
    theta1 = 2*pi*rand;
    rand1 = abs(F(x))*cos(theta1);
    rand2 = abs(F(x))*sin(theta1);
    H(x)  = abs(F(x))*cos(0);
    R(x) = complex(rand1,rand2);

end


% Her tar jeg invers fourier transformasjon av arrayen med
randomiserte faser.
% Her får jeg altså signalet som kommer fram etter å ha
randomisert fasen.
randomsignal=ifft(R,N)*N;
K = real(randomsignal);
randomsignal;

figure(3)
plot(t,K)
xlabel('Tid')
ylabel('Amplitude')
title('Signal etter randomisering av fasen.')
wavplay(K,Fs)
% Her tar jeg invers fourier transformasjon av arrayen av
nullstillt fase
Handsomsignal=ifft(H,N)*N;
Y = real(Handsomsignal);

pause(3)
figure(4)
plot(t,Y)
xlabel('Tid')
ylabel('Amplitude')

title('Signal etter nullstilling av fasen.')
wavplay(Y,Fs)
```

Her er plottene for lydfil2:

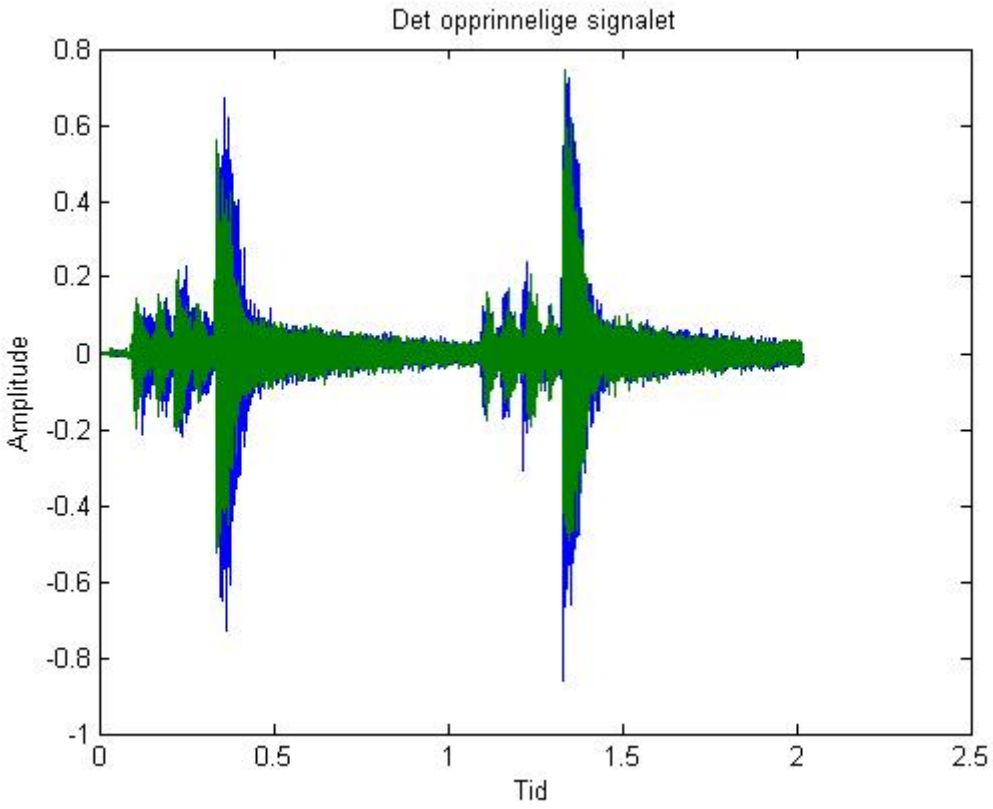


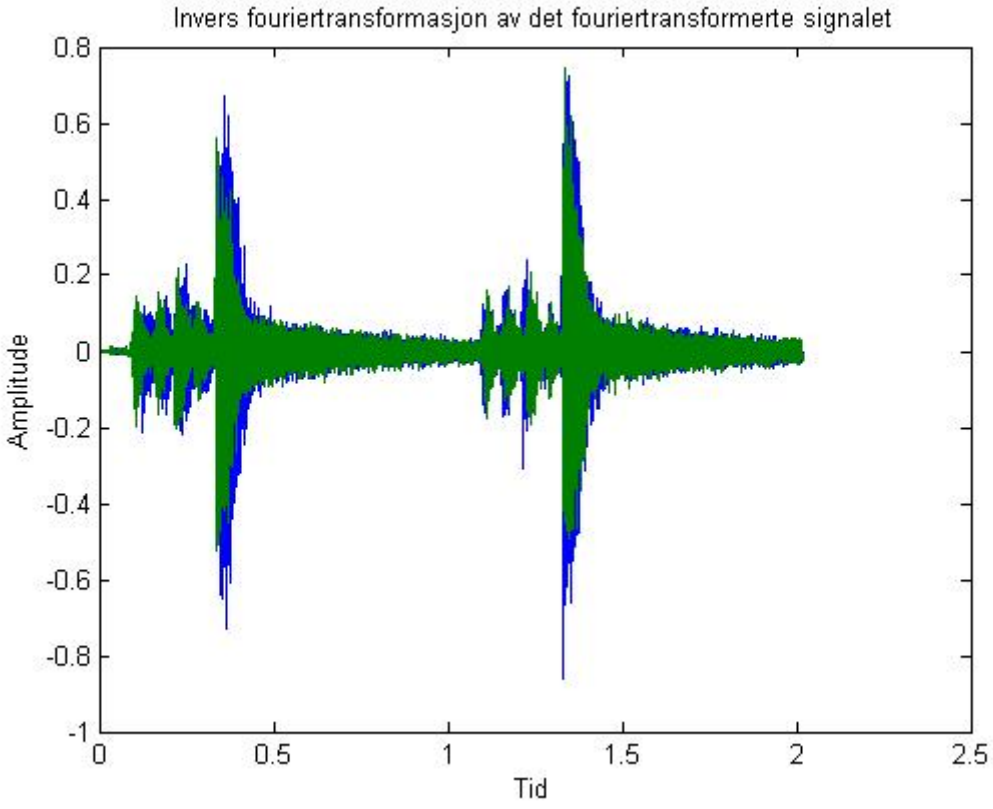




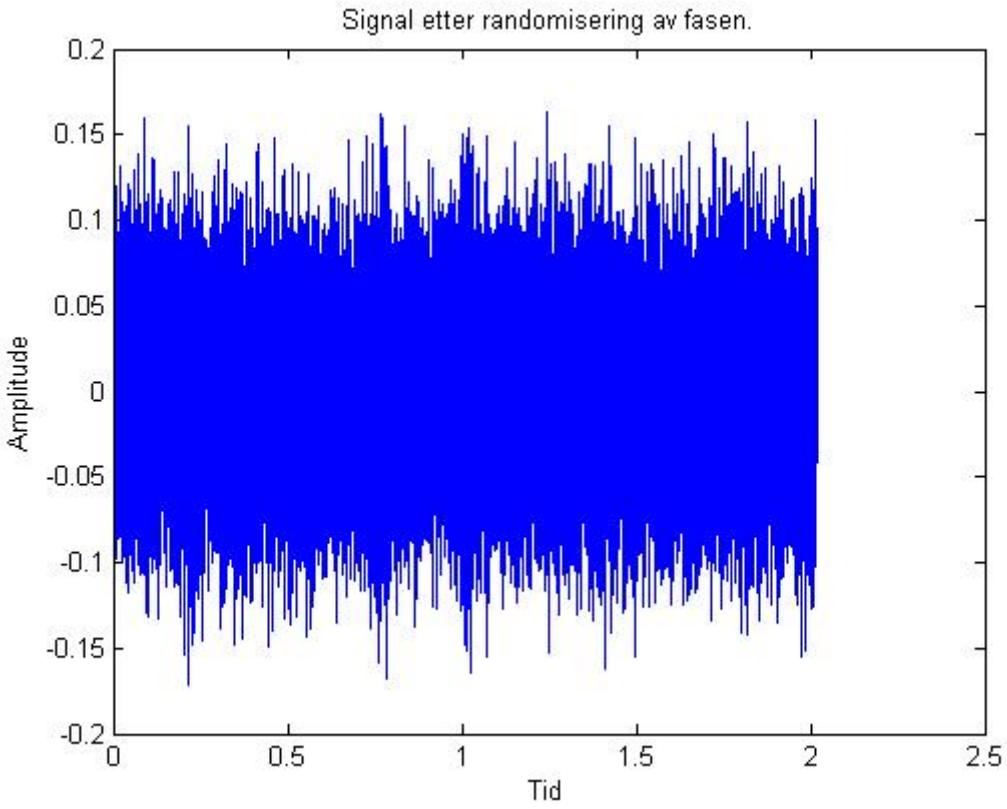


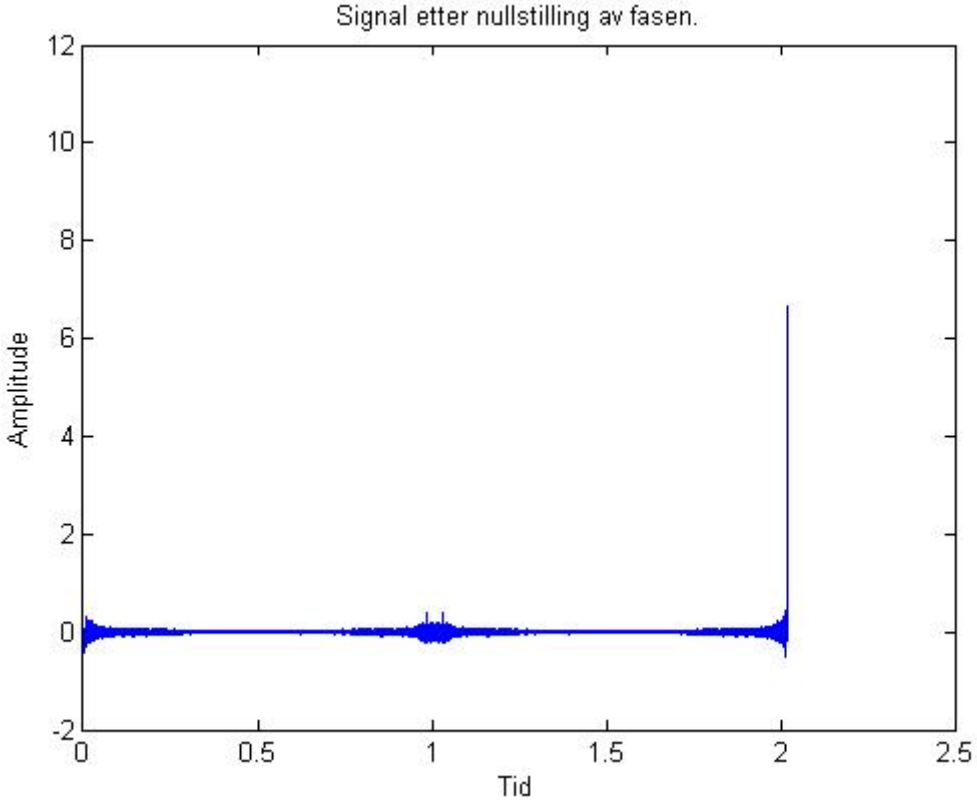
<u>Her er plottene som hører til signalet YelloBostich2:</u>


Det opprinnelige signalet
Amplitude
Tid
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5


Invers fouriertransformasjon av det fouriertransformerte signalet
Amplitude
Tid
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5

,


Signal etter randomisering av fasen.
Amplitude
Tid
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
0.5
1
1.5
2
2.5


Signal etter nullstilling av fasen.
Amplitude
Tid
12
10
8
6
4
2
0
-2
0
0.5
1
1.5
2
2.5

Konklusjon til oppgave 3.

Det vi kan konkludere med er at desto flere ledd i linear kombinasjonen som trengs for å generere et signal, desto lettere vil det være å forstyrre signalet gjennom randomisering av fase, eller nullstilling av fasen. Dette kommer fram av at lydfil2, som kan skrives som en linearkombinasjon av færre trigonometriske funksjoner, er mye lettere å kjenne igjen etter manipulering enn YellowBostich2, som må skrives som en lineærkombinasjon av langt flere trigonometriske funksjoner.