Oblig 1 FYS2130

Elling Hauge-Iversen

February 9, 2009

Oppgave 1

For å estimere kvalitetsfaktoren til basilarmembranen for ulike frekvenser har jeg laget et program som generer et rent sinussignal. Ideen er å finne den minste endringen i bølgelengden som er hørbar for et menneske. Derfor spiller jeg et signal med frekvens ν i ca et sekund for så endre signalet med en $\Delta\nu$ som er forhåndssatt. Programmet er skrevet i matlab.

```
1  fs = 22050;
2  fr = 743;
3  df = 2.4;
4
5  % Tid mellom hver sampling :
6  tid = 1.2*rand();
7  delta_t = 1/Fs ;
8  % Antal l samplinger :
9  N = tid*Fs;
10
11 % Tidsvektor :
12 t = (0:delta_t:tid);
13
14 f1 = sin(2*pi*fr*t); %tidssignal 1
15 f2 = sin(2*pi*(fr+df)*t); %tidssignal 1 + delta
16
17 s = [f1 f2]; % satt sammen til et signal som endrer seg
18 sound(s,fs); % spiller av signalet
```

Variablene 'fs', 'fr' og 'df' er henholdsvis samplingsraten, frekvensen som signalet skal ha (her satt til 743), og den Δ som frekvensen skal endres til (her satt til 2.4). Variabelen 'tid' i 6. linje bestemmer hvor lenge signalet skal spilles før frekvensen skiftes. Her har jeg lagt inn en randomfunksjon, slik at man ikke kan gjette seg til når signalet skifter. Jeg har utelatt å angi en amplitude, den blir dermed automatisk lik 1 når signalet genereres. Dette er fordi signalet ble skurret hver gang jeg satte inn en variabel i signaluttrykket. Jeg fikk bedre resultat når jeg brukte en ekstern forsterker til å endre volum på output.

Resultater og vurdering

Jeg skulle velge to frekvenser innenfor frekvensområdet 300 - 1000 Hz. Jeg valgte derfor frekvensene 450 Hz og 743 Hz. Kvaliltetsfaktoren Q kan grovt estimeres:

$$Q = \frac{\nu}{\Delta\nu} \tag{1}$$

Hvor $\Delta\nu$ er den minste hørbare frekvensendringen fra gitte frekvens Her er resultatet:

ν	$\Delta\nu$	Q
450 Hz	1.5 Hz	300
743 Hz	2.5 Hz	297.2

Disse to resultatene er så like at jeg annser målingene som korrekte. Det er for få målinger til å dra en slutning om hele basilarmembranens kvalitetfaktor, men det kan virke som at $\Delta\nu$ øker når ν øker slik at Q ligger på 300.

Oppgave 2

I oppgaven skal man finne ut hvor lang tid man må høre en frekvens, for at man skal være i stand til å si noe om tonehøyden til lyden.

Jeg modifiserte programmet i oppgave 1.

```
1  Fs = 22050;
2  fr = 800 + 100*rand(); %frkvens
3
4  % Tid mellom hver sampling :
5
6  tid1= 0.02;
7  tid2 = 1.0;
8  delta_t = 1/Fs ;
9
10 % Antal l samplinger :
11 N1 = tid1*Fs;
12 N2 = tid2*Fs;
13
14 % Tidsvektor :
15 t1 = ( 0:delta_t:tid1);
16 t2 = (0:delta_t:tid2);
17
18 %to like lydsignaler med forskjellig lengde
19 f1 = sin(2*pi*fr*t1);
20 f2 = sin(2*pi*fr*t2);
21
22 sound(f1, Fs)
23 pause(1.5)
24 sound(f2, Fs)
```

'Fs' er samplingsraten. 'fr' er frekvensen til signalet. Her har jeg lagt inn en randomfunksjon, slik at man ikke skal kunne 'huske' lyden når man kjører programmet to ganger etter hverandre. Frekvensen blir derfor liggende mellom oppgitt frekvens og frekvensen som er 100 Hz høyere (her 800 - 900 Hz). 'tid1' er hvor lang tid signalet skal spilles når man skal prøve å oppfatte tonehøyden. 'tid2' er hvor lang tid signalet skal spilles etterpå for å sammenligne oppfattet tone og faktisk tone. I de tre nederste linjene spilles signalene med en pause på 1.5 sekunder.

Resultater og vurdering

Dette er resultatene:

ν	oppfattelsestid
360 - 460 Hz	0.02 sec
520 - 620 Hz	0.02 sec
800 - 900 Hz	0.02 sec

Siden disse resultatene ga akkurat de samme resultatene på oppfattelsestid valgte jeg å prøve en utenfor mitt spesifiserte frekvensområde (300 - 1000 Hz). Jeg prøvde med frekvenser mellom 100 - 200 Hz. **Her måtte jeg sette 'tid1' til 0.5 sec for å oppfatte hvilken tonehøyde signalet var i**. Dermed kan det virke som at jo mindre frekvensen er, desto lenger må vi høre signalet før vi kan bestemme tonehøyden. Samtidig vil en frekvensendring høres lettere for lave frekvenser enn for høyere (fra oppg 1).

Basilarmembranen er bygd opp slik at hvert punkt på membranen har sin enestående egenfrekvens. Når membranen blir påført ytre svingninger med en bestemt frekvens, vil den delen av membranen som tilsvarer denne frekvensen starte å svinge. For at vi skal kunne oppfatte hvilken tonehøyde et signal er i, må membranen ha oppnådd en svingning lik den ytre svingningen. Q verdien er definert:

$$Q = 2\pi \frac{E}{t} \tag{2}$$

Q er kvalitetsfaktoren, E er energien til svingningen med en bestemt frekvens, mens t er tapet av energi pr. periode. En svingning som ikke blir påtrykt en ytre kraft vil etter en tid dø ut. Dette kan også snus, slik at vi kan si 't' er den energien en svingning 'tar opp' for hver periode fra en ytre kraft. Hvis vi nå snur på uttrykket før vi:

$$t = \frac{2\pi E}{Q} \tag{3}$$

Siden man ut i fra oppgave 1 kom frem til at Q-verdien er stabil for alle frekvenser. Dermed trenger hver eneste frekvens det samme antall perioder for å oppnå et stabilt signal. Dermed sier det seg selv at det vil ta lenger tid å oppnå et stabilt signal for en lavere frekvens.

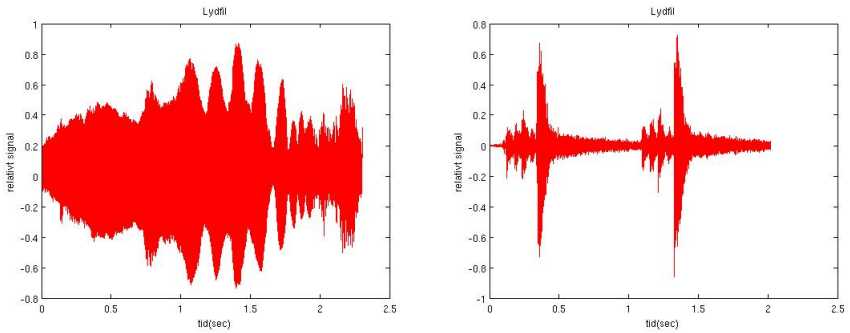
Oppgave 3

Programmet for å fourieromvende en tidsstreng:

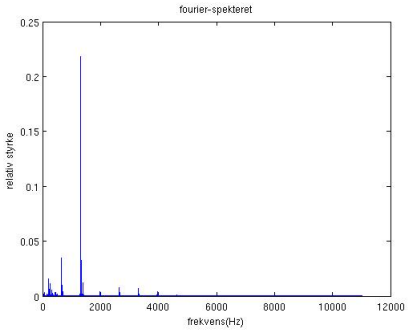
```
1 c = 'lydfil2.wav';
2 %c = 'YelloBostich2.wav';
3
4 %Leser inn fil og setter variable
5 [fil, Fs, bit] = wavread(c);
6 N = length(fil);
7 T = N/Fs;
```

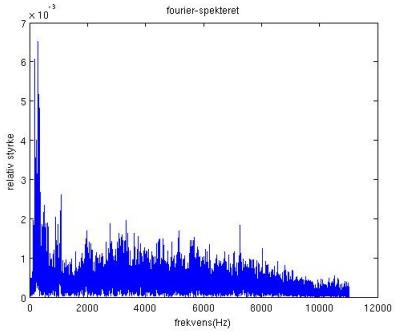
```
8
9  %verdier for plotting
10 t = linspace(0,T*(N-1)/N, N);
11 g = zeros(N, 1);
12 delta_f = 1/T;
13 fmax = N/T;
14 f = linspace(0,fmax*(N-1)/N,N);
15 g = fil(:,1);
16
17 sound(g,Fs); %Spiller av lyden
18
19 % Plotter lydsignalet
20 figure
21 plot(t,g,'-r');
22 title('Lydfil')
23 xlabel('tid(sec)');
24 ylabel('relativt signal');
25
26 %fourier trans.
27 G = fft(g,N)/N;
28
29 m = abs(G);
30 M = N/4;
31 figure
32 plot(f(1:M),2*m(1:M)); % Plotter fourierspekteret
33 title('fourier-spekteret')
34 xlabel('frekvens(Hz)');
35 ylabel('relativ styrke');
```

Variabelen 'c' representerer lydfilen. I linje 5 blir lydfilen lest inn. 'fil' blir listen med signalverdiene, 'Fs' får samplingsraten til filen. 'N' er lengden av filen, 'T' blir lengden på signalet i sekunder. Linje 10 - 15 er diverse verdier som trengs for x-aksene i plottene. sound(g,Fs) spiller av lyden. Så plottes signalet i linje 21. Signalet fouriertransformeres i linje 27. Dette frekvensspekteret plottes i linje 32. Når dette scriptet brukes på de to lydfilene **lydfil2.wav** (venstre) og **YelloBostich2.wav** (høyre) får vi ut:



Som er slik lydfilen er i opprinnelig tilstand. Frekvensspekterene blir slik:





For å omvende signalet tilbake fra frekvensspekteret:

```
% Transformerer tilbake
g2 = ifft(G, N)*N;
sound(g2,Fs); %Spiller av

% Plotter Tilbaketransformert lydsignal
figure
plot(t, g2,'-k');
title('Lyd')
xlabel('tid(sec)');
ylabel('relativt signal');
```

Dette transformerer frekvensspekteret tilbake til det opprinnelige lydsignalet.

Hvis vi randomiserer fasen før vi transformerer signalet tilbake,er det viktig å huske på at amplituden må forbli den samme. Amplituden er gitt ved:

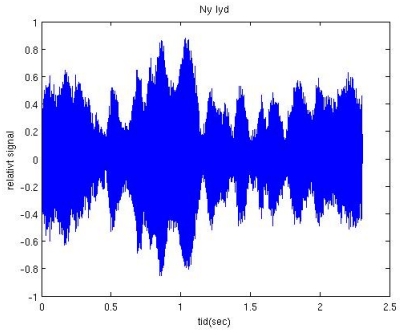
$$A = \sqrt{Re(G_i)^2 + Im(G_i)^2} \tag{4}$$

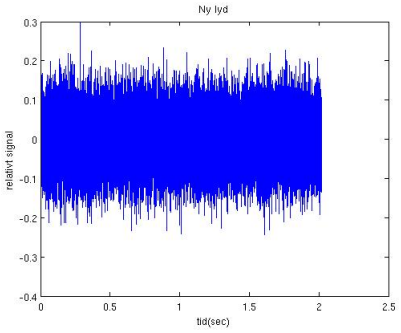
En annen ting som er viktig å huske på er at halve fourierspekteret er en speiling av seg selv. Vi må derfor sørge for at spekteret med radomisert fase fremdeles er et speilet spekter. Programmet blir da seende sånn ut:

```
%endring av fase

for i = [1:N/2];
    A = sqrt(real(G(i))^2 + imag(G(i))^2);
    phi = 2*pi*rand(); % randomisering av phi
    %phi = 0; %phi lik 0
    G(i) = complex(A.*cos(phi), A.*sin(phi));
    G(N+2-i) = complex(A.*cos(phi), -A.*sin(phi));
end

m = abs(G);
M = N/4;
```

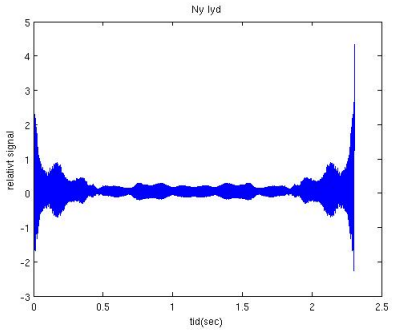
```
13
14 figure
15
16 %plotter Fourierspekteret etter faseforandring
17 plot(f(1:M), 2*m(1:M));
18 title('Nytt Fourierspekter');
19 xlabel('Frekvens [Hz]');
20 ylabel('Relativ styre');
21
22
23 g2 = real(ifft(G, N))*N;
24 sound(g2,Fs)
25
26 %Plotter nytt lydsignal
27 figure
28 plot(t, g2);
29 title('Ny lyd');
30 xlabel('tid(sec)');
31 ylabel('relativt signal');
```

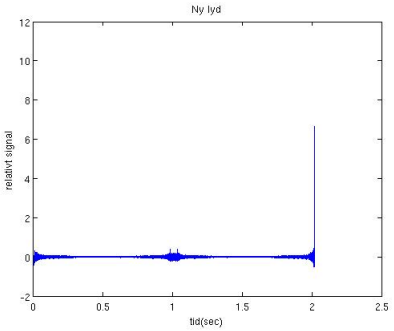
Her har vi en for-løkke som plukker amplituden til signalet, velger en tilfeldig fase, for så å putte den nye verdien tilbake både i den 'rette' og den tilsvarende speilede delen. Så plottes det nye fourierspekteret (som må forbli det samme som opprinnelig). Til slutt omvender vi spekteret tilbake til et lydsignal. Med en randomisert fase blir lydsignalene slik:





Når man i for-løkka setter 'phi' til å være 0, blir lydsignalene slik:





Resultater og vurdering

Beskrivelse av lydene

Lydfil2.wav (plottene til venstre) er et opptak av en obo som spiller en jevn men varierende tone, det er også strykere som spiller pizzicato i bakgrunnen. Når man kun fouriertransformerer og detransformerer tilbake igjen før vi ut akkurat den samme lyden.

Når vi randomiserer fasen får vi detransformerer tilbake. Det er fremdeles mulig å høre at det er en obo som spiller, men tonen høres nå mer smørt ut og det er liten eller ingen variasjon i tonen. det høres også ut som om lyden 'pulserer' dvs. lydnivået stiger og synker hele tiden. Strykernes pizzicato er helt borte.

Når vi setter fasen til å være lik 0, kan man fremdeles høre at det er en obo, men nå er lyden 'fjern'. Lydnivået begyner høyt, faller veldig raskt, har en liten bulk i volum å midten, før det stiger veldig raskt mot slutten.

YelloBostich2.wav (plottene til høyre) er et opptak av to trommevirvler i rask rekkefølge. Samme lydbilde etter å ha transformert og detransformert igjen.

Når vi randomiserer fasen blir det bare skurr. Det er ikke mulig å gjenkjenne noe av lyden fra det opprinnelige signalet.

Når fasen blir satt til å våre lik null blir det bare susing som i likhet med 'lydfil2.wav' har et høyt volum i starten, liten bulk på midten, før det blir høyt på slutten.

Det kan virke som om 'skarpe' lyder faller ut når vi randomisere fasene. Siden pizzicatoen til strykerne i den første lyden og trommene i sin helhet i dan andre, faller ut. Mens den jevne obolyden blir igjen Det er ingen tvil om at man kan oppfatte en endring i fasen.

kilder:

programmene er stort sett modifisert ut i fra utdelt program. jeg har også brukt hjelpen som er lagt ut på emnesidene for FYS2130.