FYS2130 Prosjektoppgave vår 2009, ved UiO

Wavelet-analyse
av tidsvariable signaler

av kandidat nummer 43

Innholdsfortegnelse

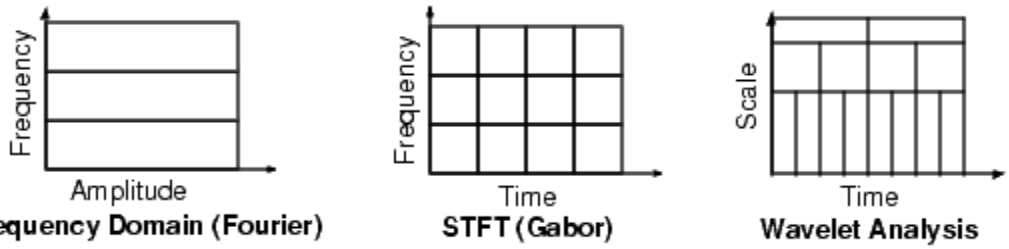
Innledning

I denne prosjektoppgaven blir wavelet-analyse introdusert og utforsket. Wavelet-analyse er en teknikk man kan bruke for å analysere mange forskjellige typer signaler; alt fra jordskjelv til finansiell informasjon til musikk. I denne rapporten skal jeg undersøke Morlet wavelets, lage et enkelt dataprogram for kontinuerlig wavelet transform, gjøre dette programmet mer effektivt ved hjelp av fourier transform og til slutt analysere diverse lydfiler. Programmet som er brukt for analysene er Matlab.

Teori

Wavelet-analyse er en teknikk lignende fourier-analyse, men med forskjellige parametere. Fourier-analyse gjør om et signal fra å se på tidsutviklingen av det til å se på hvilke frekvenser det inneholder, mens wavelet-analyse er et 3D-plot der man ser på både tidsutviklingen og hvilke frekvenser hvert tidspunkt inneholder. Fordi fourier-transformen ikke inneholder et tidsperspektiv mister vi mye informasjon hvis signalet forandrer seg over tid. Fourier-transform er best når vi ser på et repeterende signal.

Man kan tenke seg at det også er mulig å se på tidsutviklingen ved hjelp av en teknikk kalt stykkevis fourier-transform, der man deler opp et signal i mange små biter, kalt vinduer, og gjør fourier-transform på hvert vindu. Dette er en nyttig teknikk, men problemet med den er at oppløsningen på vinduene er konstant og vi vil kunne miste informasjon i signaler der frekvensen endres mye. Dette kan forbedres ved å la vinduene overlappe hverandre for så å sette sammen informasjonen etterpå.



Figur 1. Fourier-transform gir amplitude for hver frekvens, stykkevis fourier (STFT) gir frekvenser (og deres amplitude i et 3D-plot) med faste tidsintervaller. Wavelet-analyse gir en skala, som kan konverteres til frekvens,på samme måte som STFT, men man kan samtidig forandre på tidsintervallet for hvert vindu.

Teknikken som brukes i denne oppgaven er basert på at vinduene overlapper, og de blir flyttet kontinuerlig ett steg om gangen fra den ene enden til den andre, så teknikken kalles for kontinuerlig wavelet transform. Dette kjøres flere ganger med forskjellige størrelser slik at man er sikker på at man får med seg hele frekvensspekteret. Det er også mulig å forandre på steglengden slik at det ikke er en kontinuerlig forflytning, og det kalles for diskret wavelet transform, men skal ikke gjøres her. I kontinuerlig wavelet transform kan man selv velge størrelsene på vinduene helt fritt, men det kan man ikke på den diskret måten – nok en grunn for at det heter kontinuerlig.

Wavelet transform ligner ganske mye på fourier transform. Der fourier transform er basert på sinus og cosinusbølger med varierende frekvens er wavelet transform basert på små bølger, kalt wavelets, med varierende bredde og posisjon. Fourier transform er gitt ved ligningen

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

og wavelet transform er gitt ved ligningen

$$\gamma(s,\tau) = \int_{-\infty}^{\infty} f(t)\Psi^*_{s,\tau}(t)dt$$

Der f(t) er signalet som transformeres, exp(-iωt) er sinus og cosinusbølgene brukt i fourier transform og $\Psi^*_{s,\tau}(t)$ er den kompleks konjugerte av waveleten.

Waveletens bredde og posisjon varieres. Disse er gitt ved variablene s og τ. Bredden, s, kalles skalaen til waveleten og er et mål på frekvensen. Denne virker på samme måte for en wavelet som for en vanlig sinusfunksjon. Hvis man har en sinusfunksjon sin(t/s) vil frekvensen til funksjonen øke proporsjonalt med s, og funksjonen blir mer sammentrykt. På samme måte vil en wavelet bli mer sammentrykt når s øker. Når waveleten er liten vil den være følsom for høye frekvenser i signalet, mens en større wavelet vil være bedre egnet for lave frekvenser. τ er en variabel med nær sammenheng til tid, men som viser forflytningen til waveleten når waveleten flyttes bortover signalet som et vindu. Forflytning for en sinusbølge er gitt ved sin(t – n) der n er forflytningen, og det er på akkurat samme måte for wavelets som sett i ligningene nedenfor.

En wavelet kan ha veldig mange forskjellige former. Dette er en av fordelene med wavelet-analyse fordi waveletene kan tilpasses de dataene man har om man er flink. I denne oppgaven brukes en bestemt type wavelet kalt kompleks Morlet wavelet.[1]

[1] Kilder for teoridelen, og mye av resten av oppgaven er:

Matlab Help-fil (Wavelet Toolbox),
http://www.mathworks.com/access/helpdesk/help/toolbox/wavelet/index.html

Kapittel 17 i kompendiet "Wavelet analyse" av Arnt Inge Vistnes.
http://www.uio.no/studier/emner/matnat/fys/FYS2130/v09/komp13g.pdf

Vurdering av ulike Morlet wavelets (Deloppgave 1)

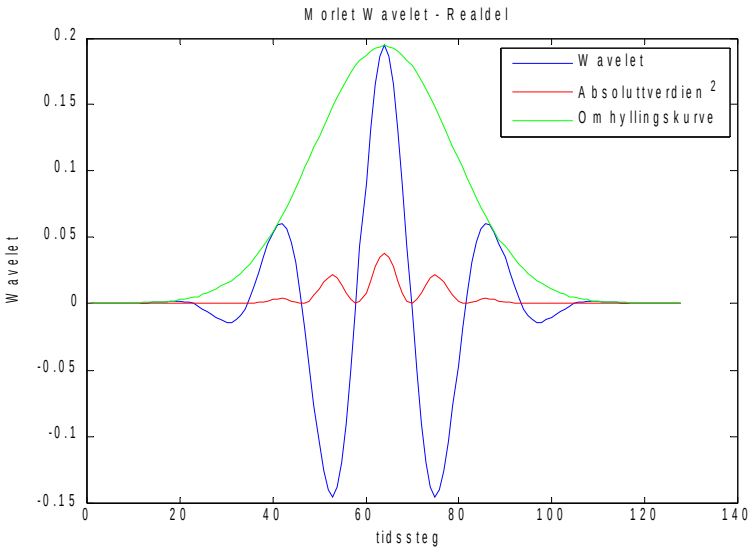
Morlet waveleten som brukes i denne oppgaven er gitt ved funksjonen

$$\Psi_{s,\tau} = \pi^{-\frac{1}{4}} e^{i\frac{\omega\tau}{s}} e^{-\frac{\tau^2}{2s^2}}$$

men omgjøres til diskret form

$$\Psi_{(s,\tau),(n'-n)} = \pi^{-\frac{1}{4}} \sqrt{\frac{dt}{s}} e^{i\omega\frac{(n'-n)dt}{s}} e^{-\frac{(\frac{(n'-n)dt}{s})^2}{2}} \quad (1)$$

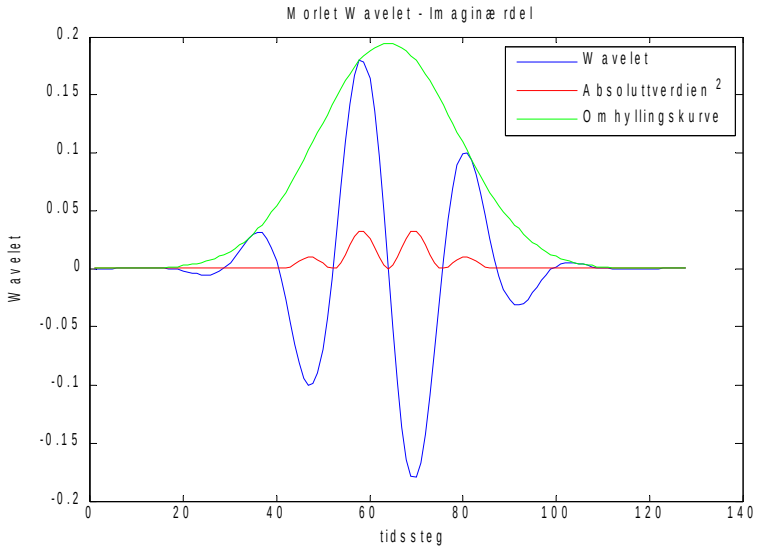
der n er en teller som gir forlytningen langs signalet og n' er plasseringen av midtpunktet til waveleten. dt er tiden mellom to forflytninger. Konstantene foran eksponentleddene er der for å normalisere waveleten slik at $|\gamma(s, \tau)|^2$ gir energien i hvert punkt. Det er i hovedsak energien som regnes ut fordi faseinformasjonen er vanskelig å beholde i en wavelet transform. Dette kan være en ulempe i forhold til fourier transform. Eksponentleddene har hver sin betydning. Det første, exp(iωt/s), gir en konstant sinusoidal bølge slik som i fourier analyse, men ganget med det andre leddet blir det avgrenset som sett i *Figur 2*. Det andre leddet er på gaussisk form og gir altså omhyllingskurven til waveleten. Det vil da si at ω er omtrent antall perioder innenfor denne omhyllingskurven.



*Figur 2. Realdelen av en kompleks Morlet wavelet med ω = 4 og s = 15*dt, dt = 0,1.*

I *Figur 2* er realdelen av en kompleks Morlet wavelet plottet i Matlab. ω er satt til å være lik 4, og som kan ses så stemmer dette overens med at waveleten har fire perioder innenfor omhyllingskurven. I *Figur 3* er den imaginære delen plottet med de samme parameterne. Det er et par forskjeller mellom disse to figurene. Den imaginære delen er litt ut av fase i forhold til den reelle delen. Dette er en positiv egenskap, for om den reelle delen såvidt blir lagt ut av fase med det

innkommende signalet slik at vi ikke får en fin maksimumsverdi, så vil den imaginære delen som følger rett etter kunne ligge riktig, og omvendt. Dette kan illustreres ved å late som om omhyllingskurven er signalet. I imaginærdelen ville da waveleten vært null der amplituden er størst, og altså ville produktet blitt null og vi hadde mistet informasjon. Derimot er waveleten størst der amplituden til signalet er størst slik at her ville vi fått med oss denne informasjonen.



*Figur 3. Kompleksdelen av en kompleks Morlet wavelet med ω = 4 og s = 15*dt, dt = 0,1.*

En annen forskjell er at imaginærdelen alltid er symmetrisk om tidsaksen. Det er ikke realdelen og det kan skape problemer fordi det er et krav at middelverdien til waveleten er lik null. Dette kommer fra admissibilitetsbetingelsen

$$\int \frac{|\Psi(\omega)|^2}{\omega} d\omega < +\infty$$

For at dette skal holde må middelverdien av Ψ(t) være lik null, og altså må integralet av den samme funksjonen også være lik null. Dette kan lett sjekkes i dataprogrammet. For ω = 4 og s = 15*dt viser det seg at integralet av realdelen er 0,0024 og for imaginærdelen er det 2,1124 x 10^{-5}. Integralet for imaginærdelen er ganske likt null, og for realdelen er det ikke så stort heller, men vi vil helst ha litt bedre presisjon.

Hvis integralet av en wavelet ikke er likt null risikerer vi å få en feil som summeres opp til en potensielt enda større feil. Dette skjer fordi vi multipliserer waveleten og signalet i hvert eneste tidspunkt og summerer produktene. Feilen vil altså da summeres opp like mange ganger som waveletens utstrekning og dette fører til at signalet kan virke sterkere eller svakere enn det det egentlig er. [2]

For å sjekke hvordan dette integralet forandrer seg med forskjellige ω og s setter jeg opp en tabell med verdier der den ene varieres og den andre holdes konstant. Fra *Tabell 1* kan man se at når ω øker blir integralet nærmere og nærmere null. Dette er logisk, fordi med flere bølgelengder

[2] Dette er svar på deloppgave 1g.

innenfor omhyllingskurven vil funksjonen være mindre avhengig av hvor bølgetoppene treffer, og bli mer symmetrisk. Som sett i *Figur 2* så treffer den første bølgetoppen ganske lavt, bølgebunnen etter det treffer ganske høyt i absoluttverdi, mens den neste bølgetoppen er enda høyere. Denne forskjellen blir mindre når ω øker. Dette kan ses i *Figur 6 og 7*. ω = 4 er på grensen og ω = 5 er ganske bra, men for å være sikker bør man bruke ω = 6 eller høyere. Det er en grunn til at vi ikke bare velger en veldig høy ω. Når ω er høy er det mange perioder innenfor omhyllingskurven til waveleten. Er ω veldig høy kan dette gjøre at vi får problemer med presisjon i Matlab fordi bølgene er for nærme hverandre og for skarpe. Når vi bruker verdiene videre er vi i hovedsak interessert i kvadratet av dem, og da blir integralet enda mindre. ω = 6 er godt nok fordi en datamaskin uansett har en presisjonsfeil, og integralet er såpass lite at det ikke får merkbare konsekvenser for analysen.

Skalaen, s, er den andre variablen. Vi setter ω = 6 og varierer s fra 2*dt til 12*dt for å finne ut hvordan integralet endrer seg med skalaen. I *Tabell 2* er dataene fra denne kjøringen av programmet. Integralet av realdelen holder seg konstant lav, mens imaginærdelen er veldig lav, men øker sakte. Konklusjonen er at selv om integralene er avhengig av skalaen, så er denne avhengigheten forsvinnende liten i forhold til avhengigheten av ω.

Integralverdier for varierende ω, s = 8*dt		
ω	Realdel	Imaginærdel
2	7,21E-001	8,06E-016
3	5,92E-002	3,08E-015
4	1,79E-003	-1,84E-015
5	1,98E-005	-2,41E-015
6	8,11E-008	2,57E-015
7	1,22E-010	1,78E-015
8	6,85E-014	-3,11E-015
9	3,68E-015	-8,46E-016
10	-2,45E-015	3,33E-015

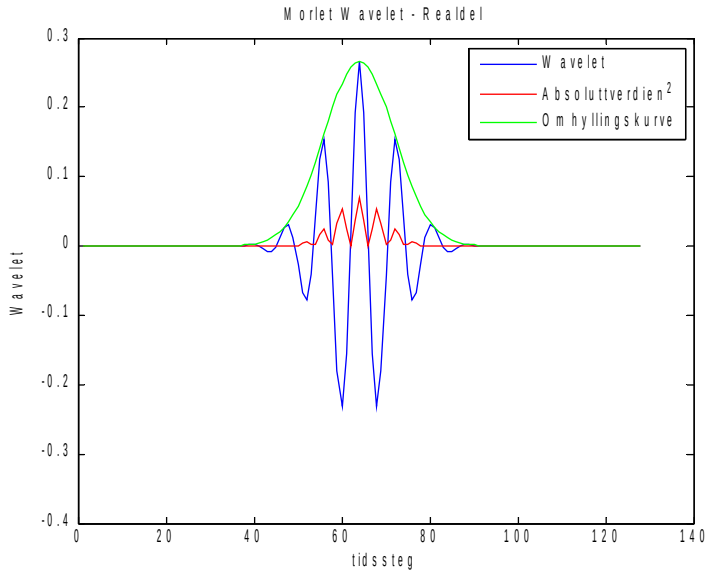
*Tabell 1. Integralverdier for varierende ω, s = 8*dt, dt = 0,1.*

Integralverdier for varierende s*dt, ω = 6, dt = 0,1		
s*dt	Realdel	Imaginærdel
0,20	4,17E-008	-1,90E-019
0,30	4,97E-008	1,37E-017
0,40	5,73E-008	-1,14E-017
0,50	6,41E-008	4,26E-017
0,60	7,02E-008	2,53E-018
0,70	7,59E-008	1,68E-017
0,80	8,11E-008	-8,02E-018
0,90	8,60E-008	2,00E-017
1,00	9,07E-008	1,13E-017
1,10	9,51E-008	-2,38E-017
1,20	9,93E-008	-1,19E-016

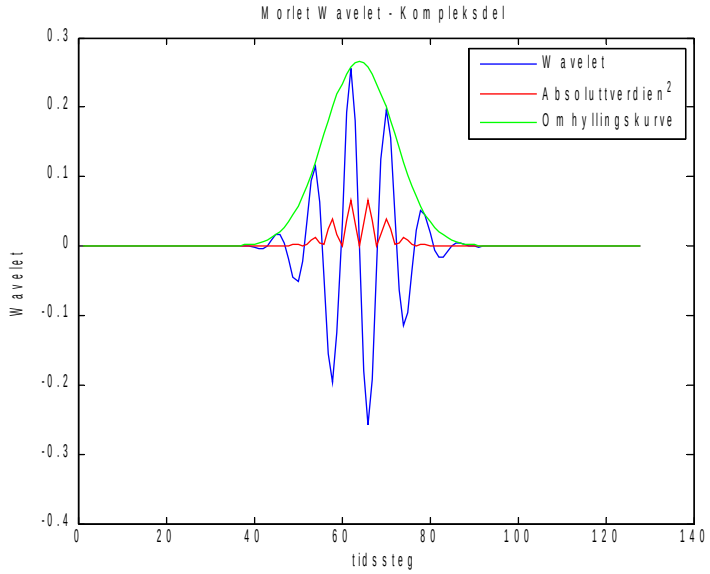
*Tabell 2. Integralverdier for varierende s*dt, ω = 6, dt = 0,1.*

Nedenfor er inkludert noen grafer med waveletene fra kjøringene ovenfor. I *Figur 8 og 9* endres skalaen og ω holdes konstant lik 6. Det er tydelig at utstrekningen av waveleten øker. Med skala 2*dt er utstrekningen omtrent 40 punkter, og med skala 12*dt er utstrekningen omtrent 80 punkter. Waveletverdiene, langs y-aksen, er forskjellige for de to. Dette er fordi waveleten er

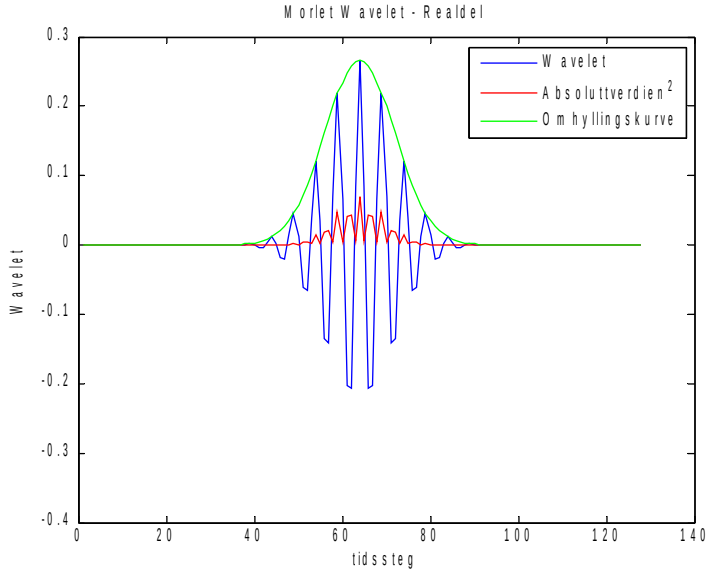
normalisert slik at energien er den samme. Normaliseringen er avhengig av en over roten av skalaen s. En viktig ting å huske på er at man må passe på å ha nok tidssteg slik at waveleten får plass. I figurene nedenfor er det brukt 128 tidssteg, som er nok for s = 12*dt, men med høyere skalaverdier bør man sikre seg at waveleten faktisk har en utstrekning mindre enn antallet tidssteg.



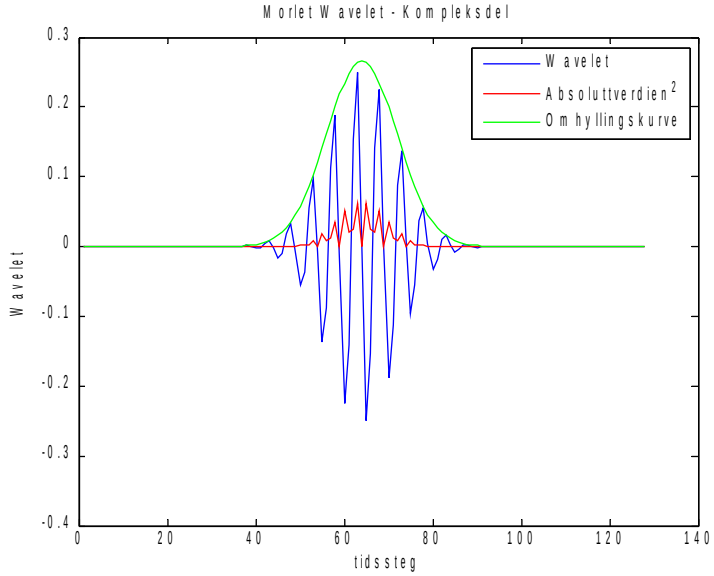
*Figur 4. Kompleksdelen av en kompleks Morlet wavelet med ω = 6 og s = 8*dt, dt = 0,1.*



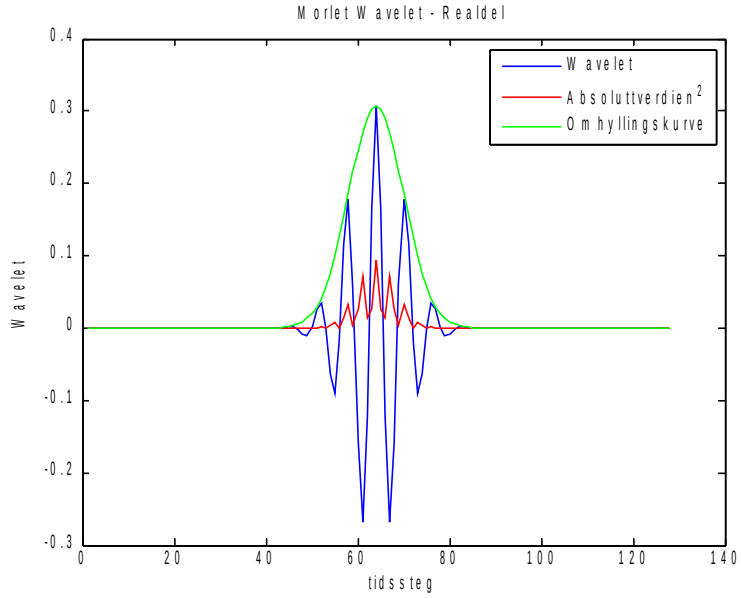
Figur 5. Kompleksdelen av en kompleks Morlet wavelet med ω = 6 og s = 8*dt, *dt = 0,1*.



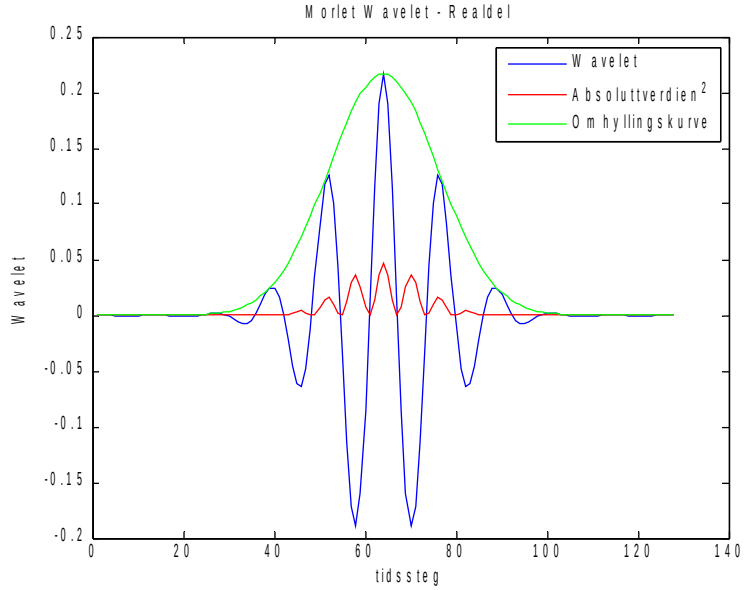
*Figur 6. Kompleksdelen av en kompleks Morlet wavelet med ω = 10 og s = 8*dt, dt = 0,1.*



*Figur 7. Kompleksdelen av en kompleks Morlet wavelet med ω = 10 og s = 8*dt, dt = 0,1.*



*Figur 8. Realdelen av en kompleks Morlet wavelet med ω = 6 og s = 2*dt, dt = 0,1.*



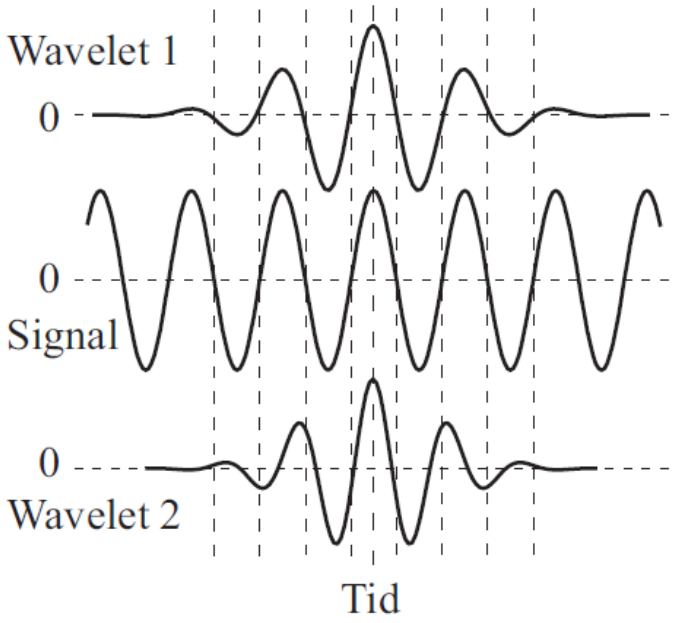
*Figur 9. Realdelen av en kompleks Morlet wavelet med ω = 6 og s = 12*dt, dt = 0,1.*

Sammenhengen mellom skala og frekvens

I wavelet-analyse brukes uttrykket skala og ikke frekvens. Jeg har til nå bare diskutert skalaen, men siden lydsignalene som skal analyseres er best forstått ved frekvenser er det viktig å finne sammenhengen mellom disse to. Wavelet-analyse er generelt og kan for eksempel brukes til å analysere korn-størrelser i krystaller, og det er ikke vanlig å snakke om en frekvens av kornstørrelser. Derfor er skala en bedre generell betegnelse enn frekvens. Sammenhengen mellom skala og frekvens kan være veldig komplisert i wavelets fordi en wavelet med en gitt skala kan tilpasses mange

frekvenser. Waveleten vi bruker, Morlet waveleten, er av en gaussisk form og periodene er uniforme innenfor dette område slik at i dette tilfellet kan vi finne en sammenheng.

For å se på hva slags frekvens en wavelet med en gitt ω og en gitt skala s tilsvarer refereres det til *Figur 10*. Når waveleten har samme periodetid som signalet vil vi få størst utslag i alle ledd. Dette gir da størst utslag på transform-verdiene, som man får ved å multiplisere signalet med waveleten og så danne en sum av produktene. Wavelet 2 vil i motsetning gi en mye mindre sum fordi noen av verdiene oppveier hverandre. Dette betyr at perioden til hver bølge innenfor waveleten bør være lik perioden til signalet, og herfra kan vi finne en sammenheng.



Figur 10. Et sinusformet signal (i midten) sammen med en wavelet med samme periodetid (øverst) og en wavlet med noe kortereperiodetid (nederst).

Det er ikke lett å dele tidsutstrekningen til waveleten med antallet perioder ω fordi waveleten ikke har en definerbar utstrekning. Det blir en subjektiv avgjørelse om hvor nær null kantene bør være før man sier at waveleten er ferdig. Derimot kan vi gjøre en analyse av uttrykket for waveleten for å finne frekvensen. Exp[i ω(n' – n)dt/s] gir den delen av waveleten som er inne i omhyllingskurven. Skalaen oppgir vi i en skalafaktor sf ganget med dt. For at denne skal være periodisk må

$$\omega*(n'-n)/sf = 2*\pi$$

når vi har gått n punkter som tilsvarer en periode. n' er også en variabel, men som en approksimasjon holder vi den i ro når vi flytter på n. Dette gir oss at (n' – n) = Δn

$$\omega*\Delta n/sf = 2*\pi$$

$$\Delta n = (2*\pi*sf) / (\omega)$$

Δn er altså en bølgelengde og ganger vi denne med dt gir det oss perioden T.

$$\Delta n*dt = T = (2*\pi*sf*dt) / (\omega)$$

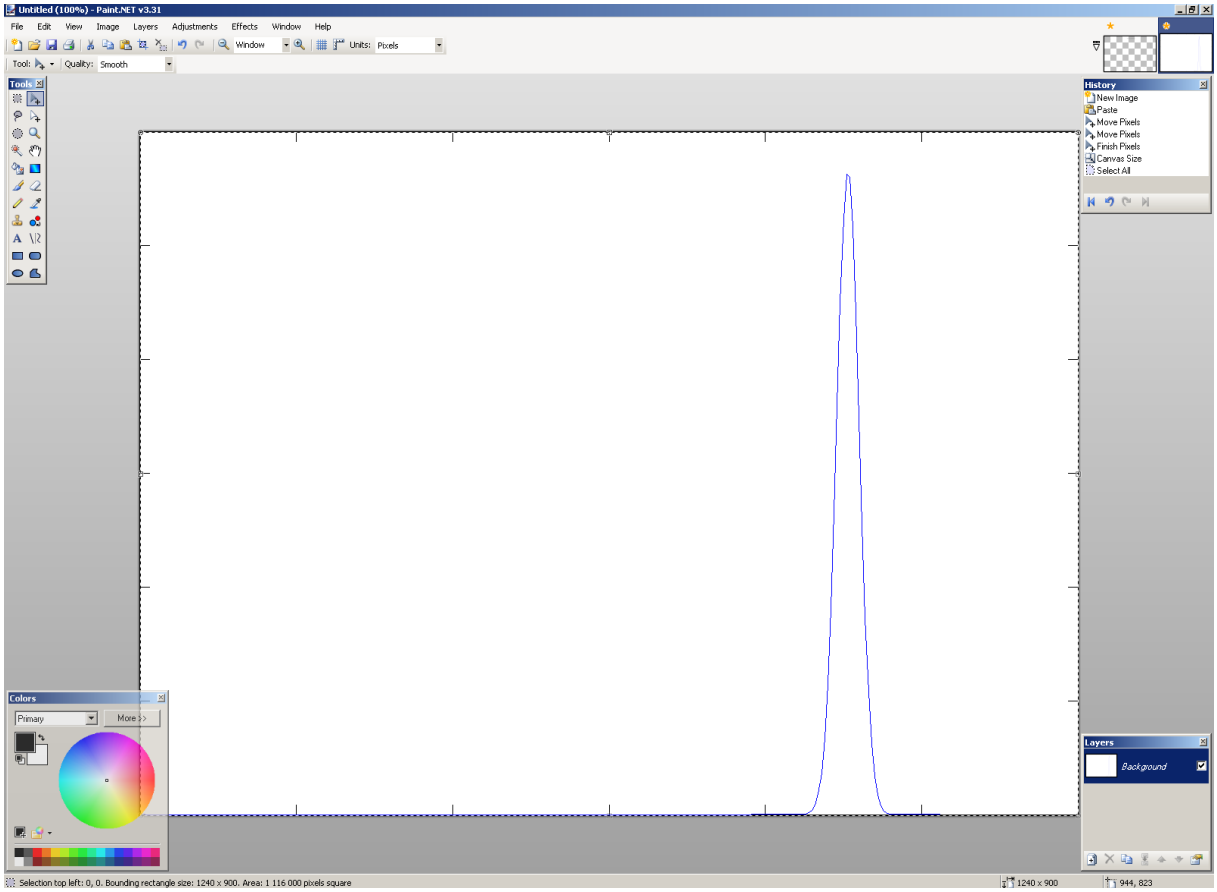
Frekvensen er 1/T slik at

$F = \omega / (2*\pi*s)$

Presisjon på frekvensoppløsningen og tidsoppløsningen

Frekvensoppløsningen og tidsoppløsningen er nært knyttet til hverandre. For å se på denne sammenhengen måler jeg den relative båndbredden av fourier transformen som et mål på frekvensoppløsningen og bredden på omhyllingskurven til waveleten som et mål på tidsoppløsningen. Det er viktig at vi bruker samme akseverdier på de to målingene slik at de kan sammenlignes.

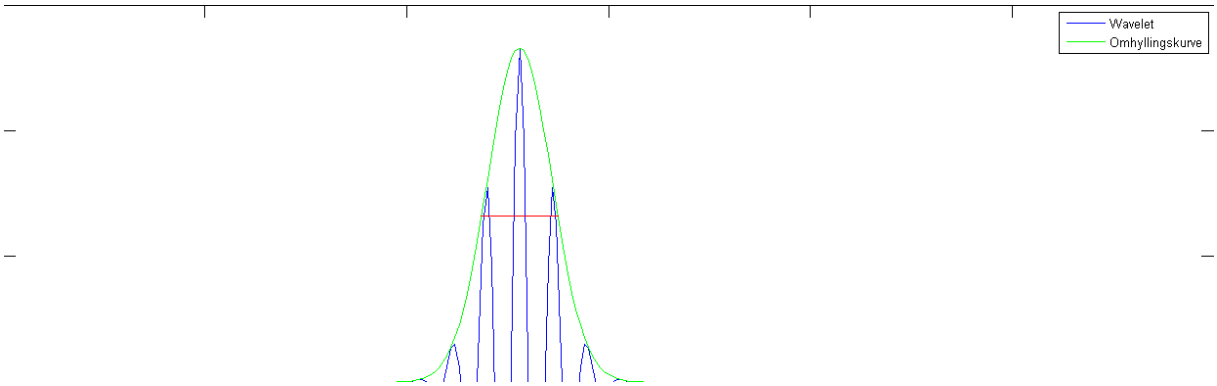
For å måle den relative båndbredden av fourier transformen plottes fourier transformen av en wavelet med ω = 6 og s = 8*dt. Deretter overførte jeg den til Paint slik at alle verdiene kunne måles med piksel-presisjon. På figuren min tilsvarte 30 verdier på x-aksen til fourier transformen 900 piksler og 300 verdier på y-aksen tilsvarte 1240 piksler. Y-aksen bestod bare av 256 punkter slik at dette tilsvarte 1058 piksler.



*Figur 11. Måling av relativ båndbredde av en fourier transform. ω = 6 og s = 8*dt, dt = 0,1.*

Toppen av kurven var på piksel 845 nedenfra. Metoden som brukes er full width at half max (FWHM) og halvveis ned fra toppen er kurven 35 piksler bred. Den relative båndbredden er da 35/1058 = 0,03308.

På samme måte blir tidsdelen, waveleten, analysert, vist i *Figur 12*. I Paint er figuren akkurat like bred som det frekvensfiguren var, det vil si at 256 punkter tilsvarer 1058 piksler. Y-aksen er denne gangen opp til 0,3 og figuren er 385 piksler høy. Toppen er på piksel 341 nedenfra. Halvveis ned fra toppen er kurven 79 piksler bred. Dette gir oss en relativ tidshalvverdi på 79/1058 = 0,07467.



*Figur 12. Måling av relativ halvverdibredde av en wavelet. ω = 6 og s = 8*dt, dt = 0,1*

Vi kan utifra disse verdiene regne ut en Δt og en Δf.

Δf = fmax * 35/1058

Δt = T * 79/1058

der T = N * dt. Vi brukte N = 256 punkter og dt var satt til å være 0,1. fmax = 1/dt. Sammenhengen mellom disse verdiene er da T * fmax = N.

Δf = (1/dt) * 35/1058 = 0,3308

Δt = N*dt * 79/1058 = 1,912

Den samme fremgangsmåten gjøres nå med en wavelet med ω = 6 og s = 16*dt. Den relative båndbredden til frekvensbildet er 17/1058, og den relative halvverdibredden av tidsbildet er 156/1058. Dette ser veldig ut som om den frekvenspresisjonen er omtrent doblet og tidspresisjonen er halvert.

Δf = (1/dt) * 17/1058 = 0,1607

Δt = N*dt * 156/1058 = 3,775

For å teste om det er en dobling/halvering av presisjon når skalaen dobles gjør jeg et nytt forsøk der ω = 6 og s = 32*dt. Her får jeg at den relative båndbredden er 9/1058 og halvverdibredden er 338. Sistnevnte er litt mer enn en dobling av den forrige verdien, men dette kan være på grunn av dårlig presisjon i målingene.

Δf = (1/dt) * 9/1058 = 0,08507

Δt = N*dt * 338/1058 = 8,178

Det ser ut som om det er en nær sammenheng mellom produktene Δt * Δf i de forskjellige målingene. Produktene er ganske like, oppsummert i *Tabell 3*, noe som antyder at når vi vil øke presisjonen til frekvensen må vi samtidig minke presisjonen til tiden.

Skala	**Δf**	**Δt**	**Δf * Δt**
8*dt	0,3308	1,912	0,6325
16*dt	0,1607	3,775	0,6066
32*dt	0,08507	8,178	0,6957

*Tabell 3. Oppsummering av verdier for Δf, Δt og produktet Δf*Δt for forskjellige skalaer. ω = 6, dt = 0,1.*

Vi har altså da en omtrentlig sammenheng Δf*Δt = 0,65. Ganger man på begge sider med Plancks konstant h får man

Δhf*Δt = 0,65h

som er det samme som

ΔE*Δt = 0,65h

der E er energien. Dette er en versjon av Heissenbergs uskarphetsfunksjon. Konklusjonen er at tidspresisjonen og frekvenspresisjonen ikke bare er begrenset av måleteknikkene vi bruker, men også av uskarphetsfunksjonen. Det er signalet selv som begrenser hva vi kan måle. Hvis signalet varer lenge kan vi stadig øke skalaen slik at waveleten blir bredere og dette ville gitt en god frekvensoppløsning helt til waveleten er like stor som signalet. Samtidig vil vi få en dårligere tidsoppløsning. ω kan også justeres for å forbedre enten frekvensoppløsningen eller tidsoppløsningen. Jo større ω er jo bedre frekvensoppløsning og jo dårligere tidsoppløsning får vi.

Dataprogrammet som er brukt i kjøringene så langt er lagt til i Vedlegg 1.

Et dataprogram for wavelet transform på intuitivt enkleste måte (Deloppgave 2)

Den mest intuitive måten å lage et dataprogram som gjør wavelet transform er å sette inn ligning (1), repetert under for enkelhets skyld, rett inn i programmet.

$$\Psi_{(s,\tau),(n'-n)} = \pi^{-\frac{1}{4}} \sqrt{\frac{dt}{s}} e^{i\omega \frac{(n'-n)dt}{s}} e^{-\frac{(\frac{(n'-n)dt}{s})^2}{2}}$$

Denne ligningen gir waveleten og man velger alle verdiene bortsett fra n som varieres. Koden jeg brukte for å beregne waveleten var

[Start Matlab-kode]

```
% Beregne waveleten
wlN = 160; % Waveletens utstrekning (med god margin for å klare skala "21")
psi = zeros(1,wlN); % Initierer waveleten psi
psi = psi + i.*psi; % Definerer psi som kompleks
wlN_mid = wlN/2; % Waveletens midtpunkt

for n = 1:wlN
    eta = (wlN_mid - n)*dt/s(s2);
    psi(n) = pi^(-1/4) * sqrt(dt/s(s2)) * exp(i*w*eta) * exp(-(eta^2)/2);
end
```

[Slutt Matlab-kode]

Fordi waveleten starter og slutter utenfor dataene legger jeg til 0-verdier både foran og bak på dataene. Fordi produktet blir null blir summen null og dataene blir ikke påvirket annet enn at de blir dempet i kantene (som de ville blitt uansett).

[Start Matlab-kode]

```
y_pad = padarray(y,[(wlN_mid-1) 0]); % Legger til wlN_mid-1 nuller før og
etter
```

[Slutt Matlab-kode]

Nå er alt klart for å gjøre selve wavelet transformen. Transformen gjøres ved å multiplisere waveleten med dataene (y) og summe alt opp, waveletens første element skal ganges med dataenes første element, helt til waveletens siste element ganges med dataenes siste element.

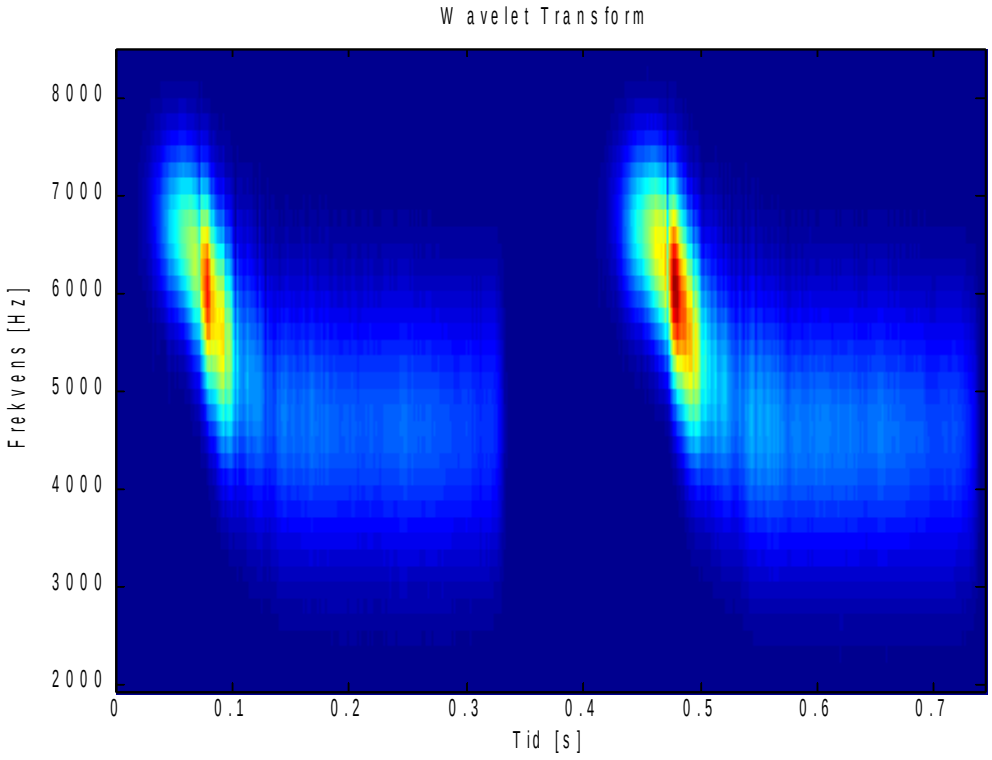
[Start Matlab-kode]

```
% For å ikke regne ut dette hver gang løkken kjøres
start_m = (wlN_mid);
stop_m = length(y_pad)-(wlN_mid-1);
start_n = -(wlN_mid-1);
stop_n = (wlN_mid);

for (m = start_m:stop_m)
```

```
    o = 0;
    for (n = start_n:stop_n)
        o = o + 1;
        wl_sum(s2,m+start_n) = y_pad(m+n) * psi(o) + wl_sum(s2,m+start_n);
    end

end
```

[Slutt Matlab-kode]

Alt dette kjøres flere ganger med ulike skalaer helt til vi får en matrise som er delt i skalaer den ene veien og tid den andre veien. Indeksen s2 i koden ovenfor viser til forskjellige skalaverdier. Hele programmet er lagt til i Vedlegg 2. Programmet brukte 117 sekunder med 40 forskjellige skalaverdier og 2^15 punkter. Det er ganske tregt å gjøre denne utregningen steg for steg og i neste seksjon gjøres dette på en mer effektiv måte. *Figur 13* viser wavelet transformen av lydfilen Granmeis2.wav, som er fuglekvitring med høy frekvens.[3] Det er usikkert om frekvensverdiene på skalaen til *Figur 13* er riktige. Formelen brukt er den utledet i seksjonen "Sammenhengen mellom skala og frekvens", men stemmer ikke overens med frekvensene fra fourier transformen som vist i neste seksjon.



*Figur 13. Wavelet transform av lydfilen Granmeis2.wav. ω = 6 og s = 8*dt til s = 21*dt i 40 lineære steg. Lydfilen består av 2^15 punkter.*

[3] Alle lydfilene brukt i denne oppgaven kan bli lastet ned fra http://www.uio.no/studier/emner/matnat/fys/FYS2130/v09/prosjektoppgavenV09.html

Et mer effektivt dataprogram for wavelet transform (Deloppgave 3)

For å gjøre programmet mer effektivt kan man kombinere wavelet transform med fourier transform. Vi benytter da en egenskap ved fourier transform som er slik at transformen av produktet av to funksjoner på samme tidsintervall er lik produktet av transformen av hver av dem separat.

$$h(t) = (f * g)(t)$$

$$H(\omega) = F(\omega) \cdot G(\omega)$$

Det vil si at vi kan ta fourier transformen av signalet og gange det med fourier transformen av waveleten for å få den fourier transformerte av begge. Nå har vi frekvensspekteret for disse to funksjonene ganget sammen, og for å få tidsspekteret kan vi ta den invers fourier transformerte av dette uttrykket. Da får vi ut samme informasjon som med wavelet transformen. Dette forutsetter at vi har en analytisk uttrykk for fourier transformen av waveleten, og det har vi i denne oppgaven fått oppgitt.

Det er flere andre ting man kan gjøre for å forbedre programmet i forrige del. En intuitiv ting er at man kan gjøre om skala/frekvens-aksen til å være logaritmisk. Dette er i samsvar med at mennesker oppfatter frekvenser logaritmisk. For å lage logaritmiske akser skriver jeg skalaen slik:

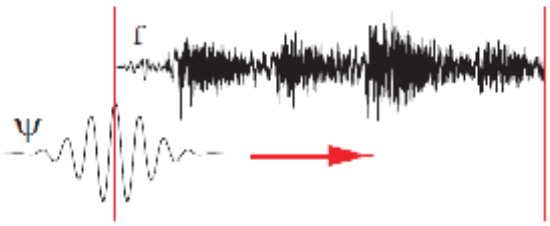
s = s0*2.^((0:sN-1)*dds)

Der s0 er en startskalaverdi og (0:sN-1) tilsvarer en vektor som går fra 0 til sN-1. 1/dds bestemmer hvor mange oppdelinger vi skal ha hver oktav. Antall skalaer skriver jeg nå som

sN = fix(11/dds)

Der tallet 11 kan forandres og gir antall oktaver. Det er dessverre komplisert å få en logaritmisk skala på plottefunksjonen imagesc i Matlab, så jeg har valgt å fjerne aksene.

En annen forbedring er at jeg legger inn en usikkerhetskurve kalt Cone of Influence (COI) i plottet. Når waveleten ganges med signalet vil deler av waveleten til tider ligge utenfor signalet. Dette medfører en demping av verdiene på starten og slutten av signalet.



Figur 14. Når waveleten først settes mot signalet er halvparten av den utenfor. Dette medfører at de første verdiene blir dempet.

Måten jeg har laget denne kurven på er ved å først beregne bredden på waveletene ved hver skalaverdi, for deretter å legge dette inn i den opprinnelige matrisen ved de lengdene halvparten av bredden av waveletene tilsvarer. Fordi jeg legger dette inn sammen med de opprinnelige dataene er det viktig at jeg ikke legger inn verdier som gjør at skalaen til amplituden til frekvensene blir feil. Derfor gjør jeg slik at verdiene får samme verdi som maksverdiene som allerede er i matrisen, og linjene som betegner usikkerhetsområdet blir da av samme farge som maksverdiene.

[Start Matlab-kode]

```
% Cone of Influence (COI)
approx = 1000; % Waveletens utstrekning slutter når verdien av den er
1/approx av maks
wl_width = ((s/dt) * 2*sqrt(2*log(approx)));
wl_width_vector = zeros(sN,N);
max_fboth = max(max(plot_fboth));

% COI shading all values outside
for s2 = 1:sN
    for n = 1:3:fix(wl_width(s2)/2)
        wl_width_vector(s2,n) = max_fboth - plot_fboth(s2,n);
        wl_width_vector(s2,N-n) = max_fboth - plot_fboth(s2,N-n);
    end
end
```

[Slutt Matlab-kode]

Man kan beregne waveletens utstrekning ved å se når omhyllingskurven er tilnærmet lik null. I koden ovenfor har jeg satt at tilnærmet lik null betyr 1/1000 av waveletens maksimumsverdi. Waveletens maksimumsverdi er når (n' – n) = 0, da er exp(0) = 1. Hvis waveleten har en bredde n er midtpunktet n' = n/2. Vi kan da sette inn i uttrykket for omhyllingskurven og at verdien skal være 1/1000 av maksimumsverdien som er 1.

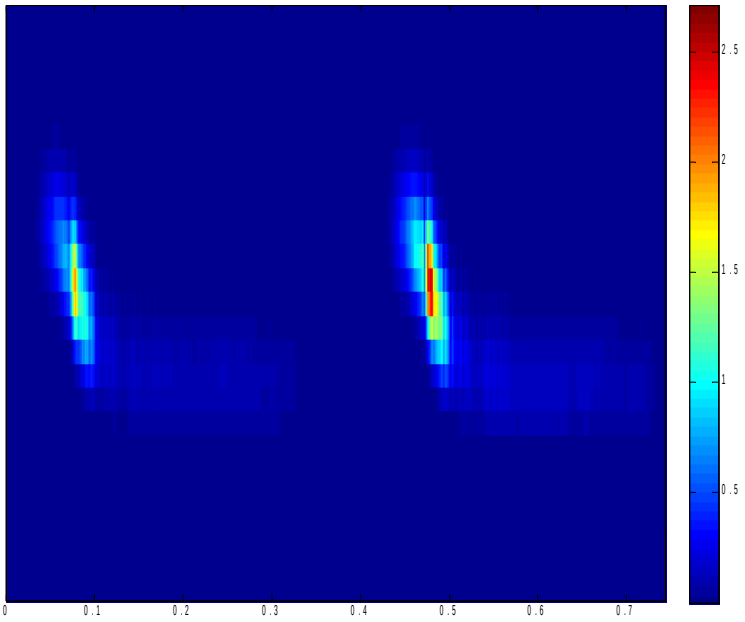
$\exp[-((n/2 - n)*dt/s)^2/2] = 1/1000$

$((n/2 - n)*dt/s)^2 = 2 * \ln(1000)$

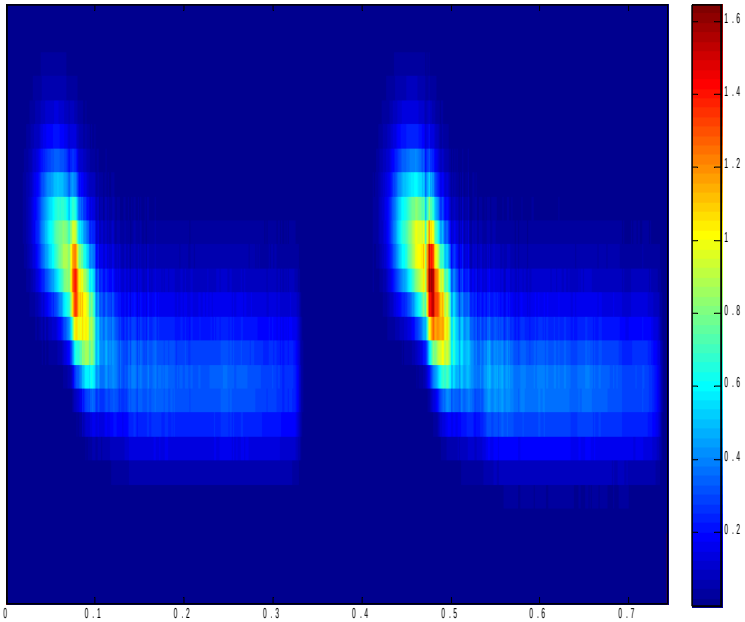
$n = (s/dt) * 2\sqrt{(2)}\sqrt{(\ln(1000))}$

Dette er uttrykket som er brukt i koden.

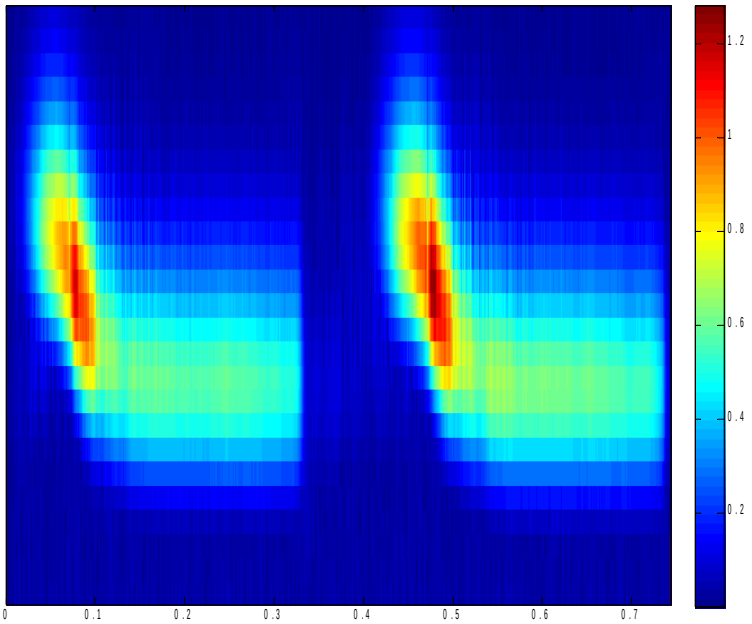
Det er vanlig å plotte wavelet power spekteret når man bruker wavelet transform. Wavelet power spekter er transformverdiene kvadrert. I Matlab kan dette gjøre at vi mister informasjon om de lavere frekvensene på grunn av graderingen av farger. Under er vist tre figurer med forskjellige funksjoner og man kan klart se at det er en fordel, i hvert fall i dette tilfellet, å se på enten verdiene som de er, eller kvadratroten av de. Fordelen med å se på kvadratet er at det blir mindre støy. COIen ses ikke i disse plottene fordi den ikke er betydningsfull i så høye frekvenser som avbildet her. Aksene er logaritmiske.



Figur 15. Wavelet power spekter, verdiene kvadrert.



Figur 16. Verdiene er plottet som de kommer ut av transformen.



Figur 17. Kvadratet av verdiene av transformen.

Mulige forbedringer av koden

Dataprogrammet jeg har skrevet er ikke optimalt. Den viktigste forbedringen er kanskje å få på riktige akser med frekvensverdier. Dette viste seg å være veldig vanskelig å få til i Matlab, men det skal være mulig. Det beste hadde vært om man kunne valgt frekvensene man vil se på og ikke forandre på skalaverdiene som man må nå. Da kunne man sett på fourier-spekteret og valgt ut det frekvensområde som er vist der.

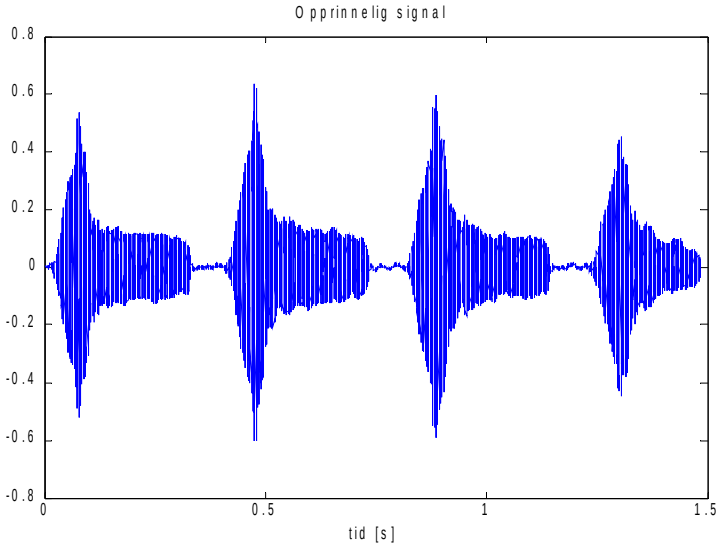
Det er mulig å lage waveleten uten å bruke for-løkker, dette ville få programmet til å gå fortere. Dette programmet var allikevel mye kjappere enn det forrige. Dette programmet brukte 5 sekunder med 40 forskjellige skalaverdier og 2^15 punkter, mens det forrige brukte 117 sekunder. Dette er mye om man trenger et slikt program og ikke har all verdens tid. Koden for COI-delen er heller ikke ideell. Den burde ha vært lagt i en vektor og så plottet oppå verdiene istedenfor å bli satt sammen med verdiene. Den er heller ikke så veldig rask på grunn av beregninger jeg måtte gjøre for å få satt dem sammen.

Analyse (Deloppgave 4)

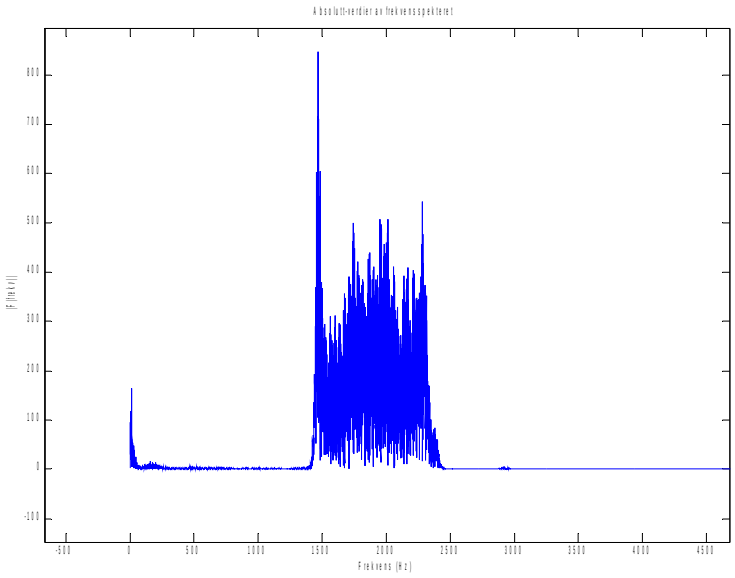
I denne delen av oppgaven skal jeg analysere fire lydfiler og se på detaljer det ikke er mulig å se ved fourier analyse. Disse fire lydfilene er Granmeis2.wav, GrexVocalis3.wav, havbolge2.wav og fagott1.wav. På begynnelsen av hver analyse viser jeg et oversiktsbilde over wavelet analysen der de samme parameterne er brukt i hvert bilde. Detter er for å vise hvilke frekvenser som finnes i signalet og for å vise usikkerhetskurven (COI). Det er viktig å vite om det er en usikkerhet forbundet med de frekvensene som blir analysert.

Granmeis2.wav

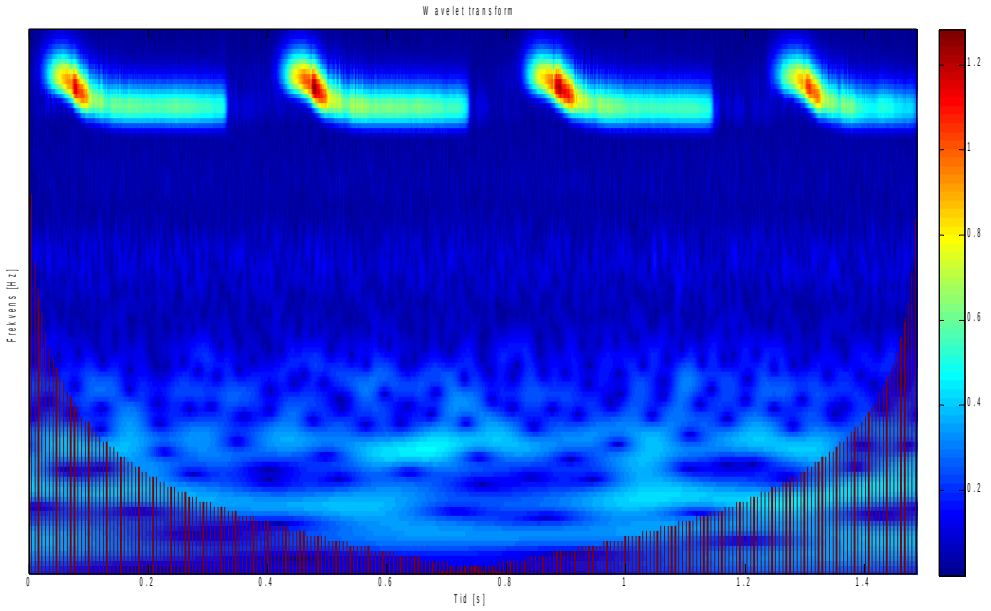
Dette er lydfilen som er brukt som eksempel tidligere i denne oppgaven. Den består av en fugl som kvitrer med korte plystrelyder i korte perioder. Tidsbildet, fourier transformen og wavelet transformen er plottet i *Figur 18, 19 og 20*.



Figur 18. Opprinnelig signal av Granmeis2.wav.



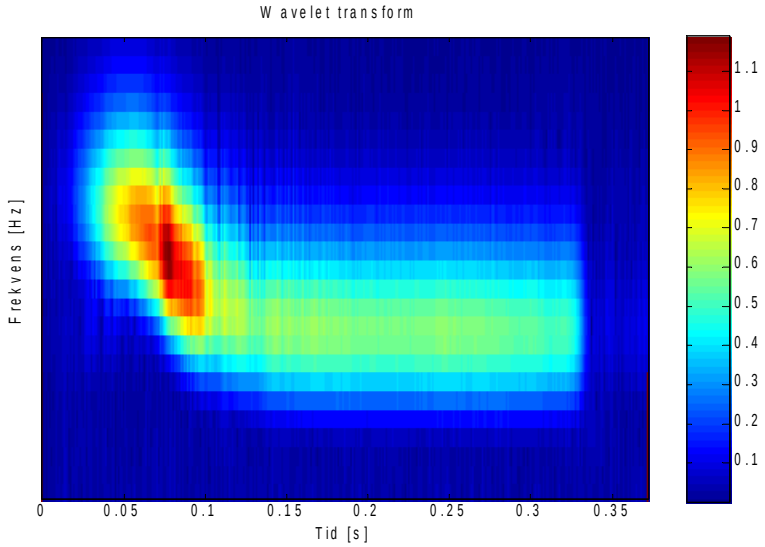
Figur 19. Fourier transform, zoomet inn på aktuelle verdier.



*Figur 20. Oversiktsbilde over wavelet transformen til Granmeis2.wav. N = 2^16 punkter, ω = 6, s0 = 5*dt og skalaen er 11 oktaver, kvadratroten av verdiene er plottet.*

I tidsbildet i Figur 18 ser vi at signalet er periodisk og at intensiteten minker innenfor hver bit. I frekvensspekteret kan vi se at frekvensene er veldig samlet uten rom imellom og at det er mest av den lavere frekvensen. Dette kan vi også se med wavelet analysen, men der kan vi også se at frekvensen endrer seg fra en høy frekvens til en lavere frekvens samtidig som intensiteten minker. Variasjonen av tiden og frekvensen samtidig får vi altså ikke fra de to andre figurene. Dette ser man

tydligere når vi zoomer inn på et av plystringene i Figur 21. Da er ω lik 6 for å bedre tidsoppløsningen for et så kort signal.

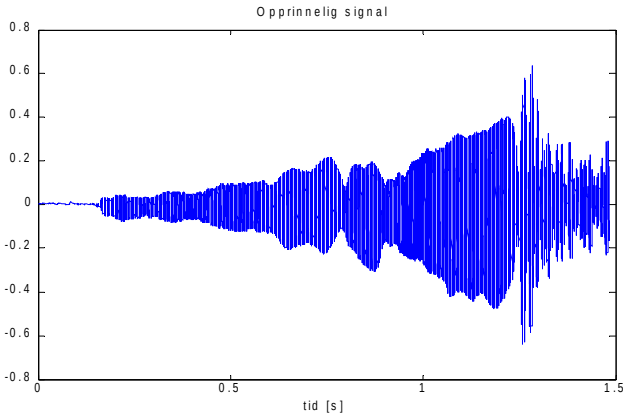


*Figur 21. En fugleplystring. N = 2^14 punkter, ω = 6, s0 = 5*dt og skalaen er 2,5 oktaver, kvadratroten av verdiene er plottet.*

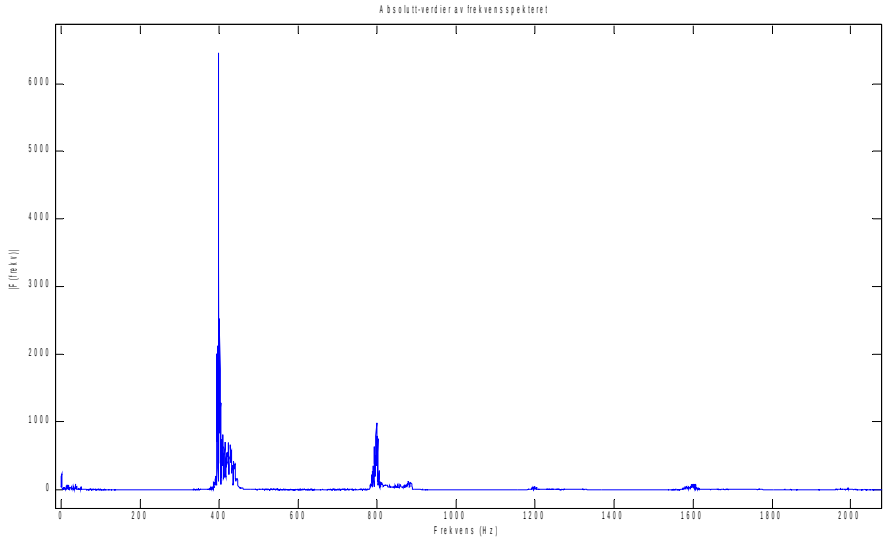
I følge frekvensspekteret skulle man tro at den laveste frekvensen var den med høyest intensitet, men dette stemmer ikke i følge wavelet analysen. Det frekvensspekteret viser er hvor mye det er av hver totalt, og det er umulig å vite at den lave frekvensen med lav intensitet går i lenger tid en den høye frekvensen med høy intensitet uten å se på wavelet transformen.

GrexVocalis3.wav

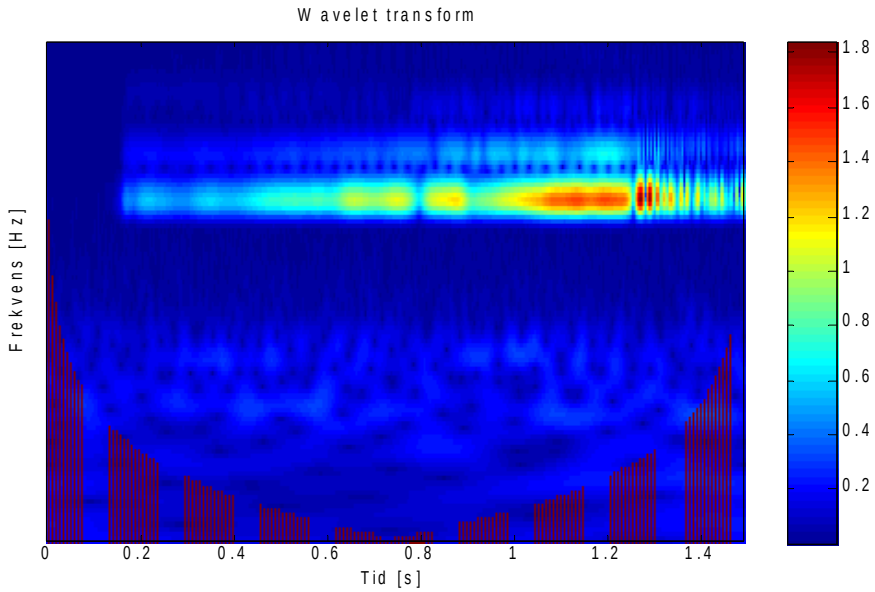
GrexVocalis3.wav er en veldig lys syngende stemme. Fra tidssignalet får vi vite at amplituden varierer litt, og ganske mye på slutten av signalet. Av frekvensspekteret får at syngingen har en høy amplitude på en lys frekvens på 400Hz og at det er overtoner på 800 Hz og såvidt noe på 1600 Hz.



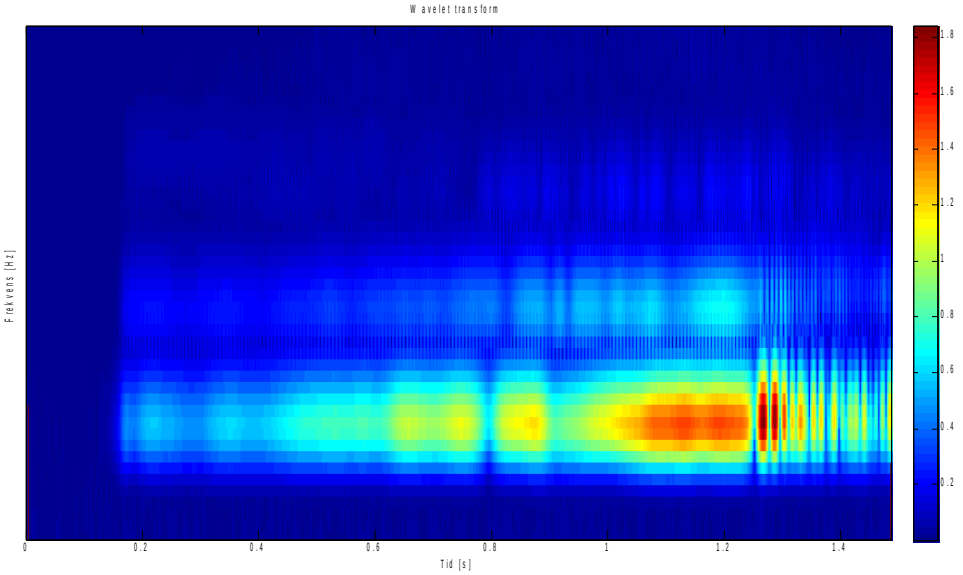
Figur 22. Opprinnelig signal av GrexVocalis3.wav.



Figur 23. Fourier transform, zoomet inn på aktuelle verdier.



*Figur 24. Oversiktsbilde over wavelet transformen til GrexVocalis3.wav. N = 2^16 punkter, ω = 6, s0 = 5*dt og skalaen er 11 oktaver, kvadratroten av verdiene er plottet.*

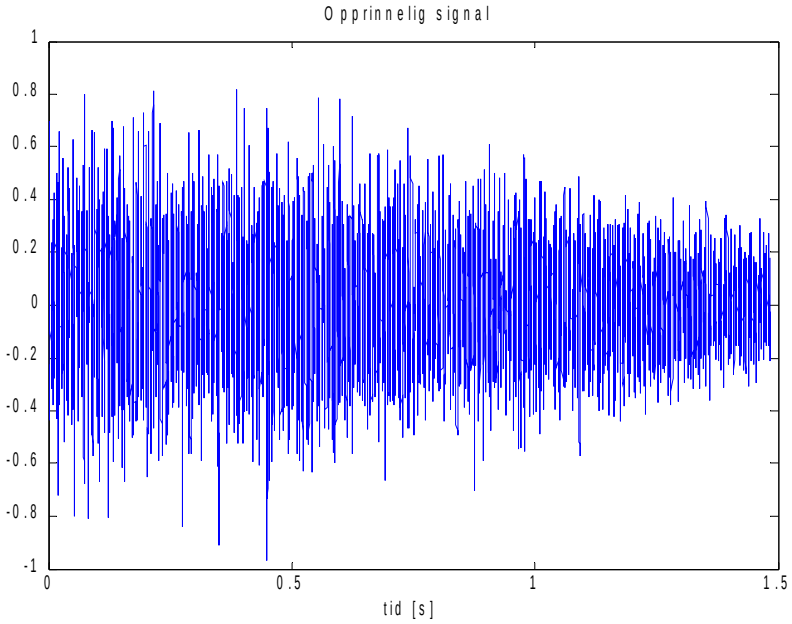


Figur 25. Begynnelsen av GrexVocalis3.wav, zoomet inn. N = 2^16 punkter, ω = 6, s0 = 5*dt og skalaen er 4,5 oktaver, kvadratroten av verdiene er plottet.

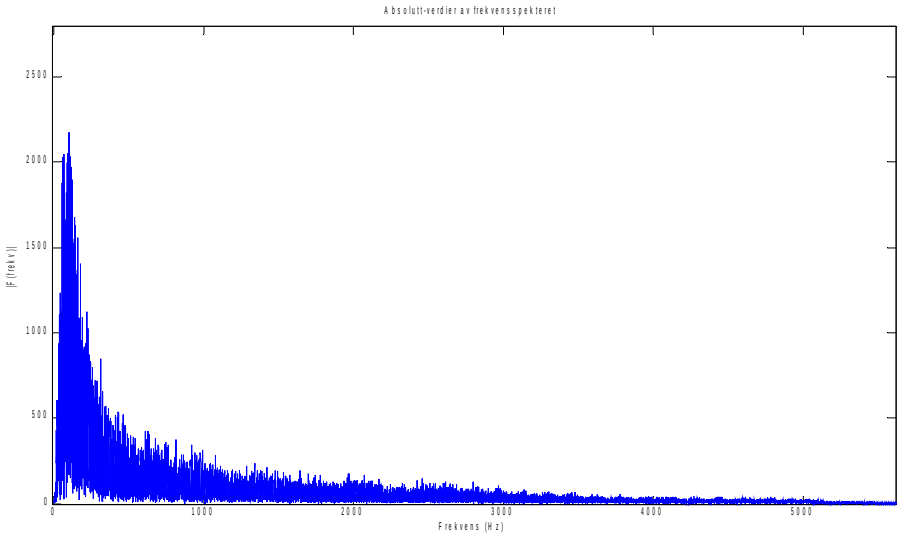
Skalaen jeg bruker i dataprogrammet er logaritmisk og er delt opp i oktaver med 10 skalaer i hver oktav. Det er interessant å telle antall ruter mellom de tre frekvensene; det er akkurat 10 ruter mellom hver. Denne sangerinnen har en ganske perfekt stemme. Det at det er overtoner her kunne man ha gjettet seg til, men ikke visst sikkert, fra fourier transformen. Det er greit å få det bekreftet at de forskjellige frekvensene forekommer samtidig. En annen ting man bare kan finne ut fra wavelet analysen er at intensiteten for hver av frekvensene øker samtidig, slik at det er ikke av og til en annen frekvens er lettere å høre. Fordi det ikke er noe særlig endring i frekvensen til dette signalet er ikke waveleter så nødvendig som for andre signaler. Derimot er det en veldig fin måte å få alle detaljene på en gang, og at sigalet faktisk er slik man antar det er fra tidsbildet og frekvensspekteret.

havbolge2.wav

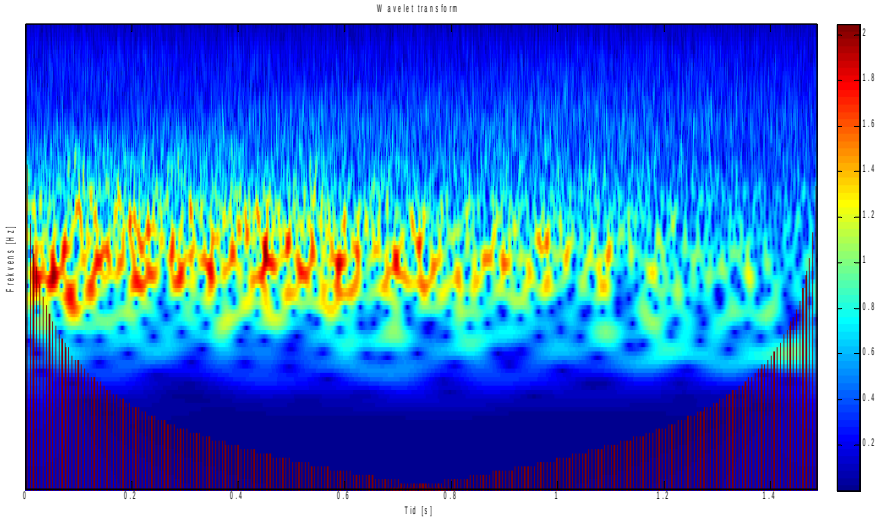
havbolge2.wav er som navnet tilsier, brusingen fra en havbølge. Dette signalet har en veldig ujevn lokal intensitet, men globalt avtar intensiteten. Frekvensspekteret viser at det er flest lave frekvenser, fra 40 Hz til 200 Hz, men med frekvenser helt opp til 7000 Hz. Det frekvensspekteret ikke viser er hvordan veldig mange av frekvensene forekommer samtidig. Det viser heller ikke at det er mange «frekvenshull» ved hvert tidspunkt, frekvensene er ikke jevnt fordelt utover. Begge disse sidene av signalet kan man se med wavelet-analysen. Signalet er veldig kaotisk, og selv på den frekvensen med høyest intensitet er det mange områder uten lyd i det hele tatt.



Figur 26. Opprinnelig signal av havbolge2.wav.



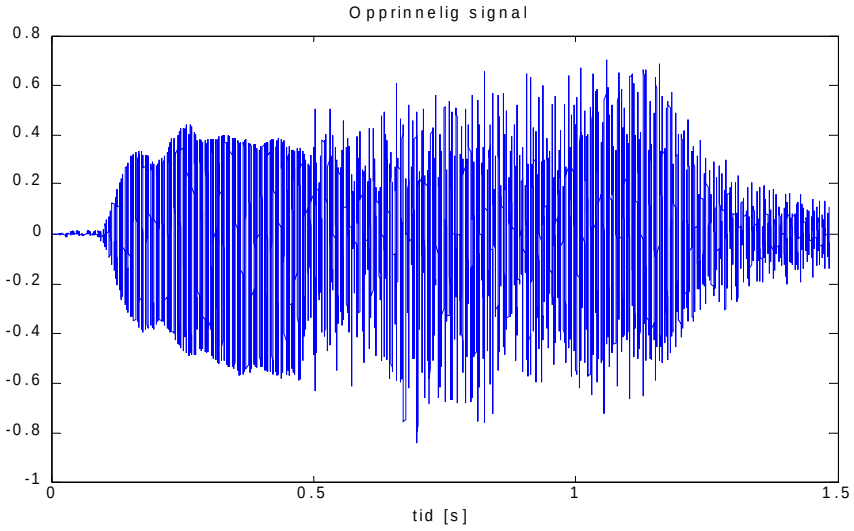
Figur 27. Fourier transform, zoomet inn på aktuelle verdier.



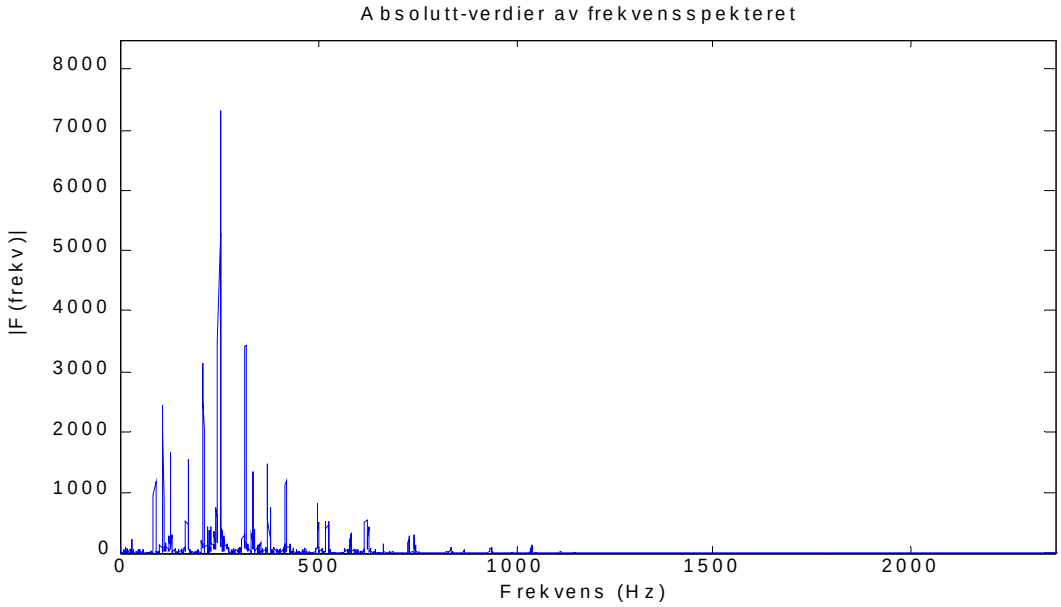
*Figur 28. Oversiktsbilde over wavelet transformen til havbolge2.wav. N = 2^16 punkter, ω = 6, s0 = 5*dt og skalaen er 11 oktaver, kvadratroten av verdiene er plottet.*

fagott1.wav

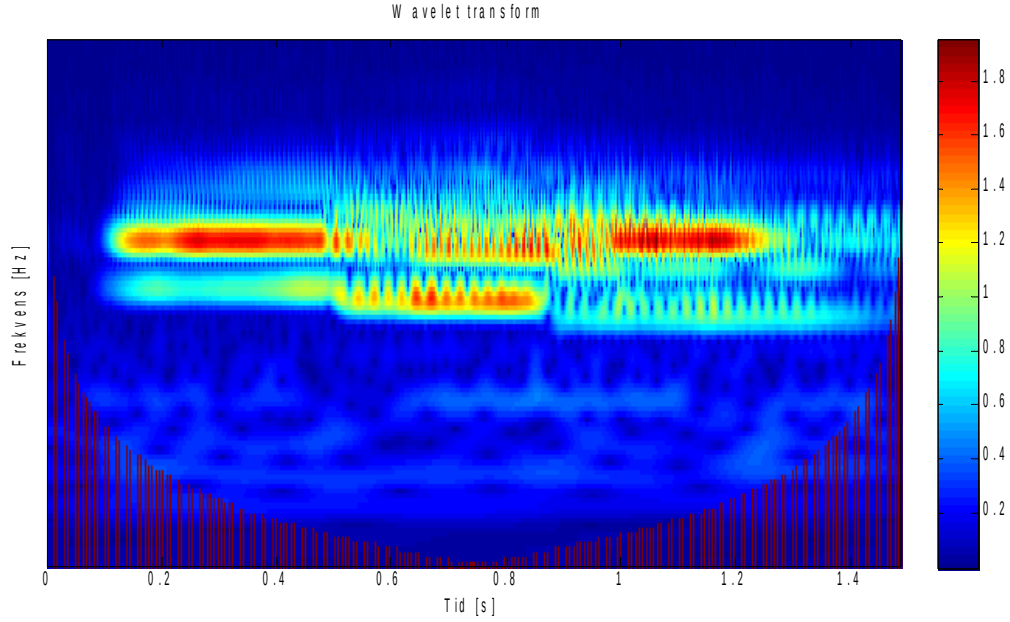
fagott1.wav er et instrument som spiller lyder med ganske lave frekvenser. Man hører at det er to forskjellige noter. Først en note, så en litt lavere, og så den første noten igjen.



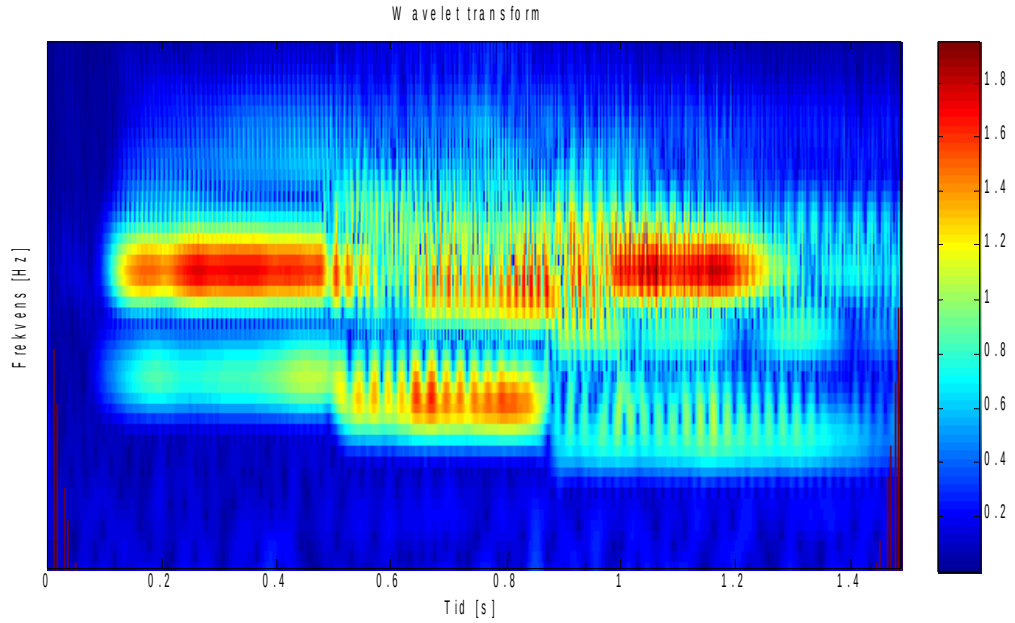
Figur 29. Opprinnelig signal av havbolge2.wav.



Figur 30. Fourier transform, zoomet inn på aktuelle verdier.



*Figur 31. Oversiktsbilde over wavelet transformen til fagott1.wav. N = 2^16 punkter, ω = 6, s0 = 5*dt og skalaen er 11 oktaver, kvadratroten av verdiene er plottet.*



*Figur 32. Begynnelsen av fagott1.wav. N = 2^16 punkter, ω = 6, s0 = 20*dt og skalaen er 5 oktaver, kvadratroten av verdiene er plottet.*

I tidsbildet i *Figur 29* ser vi at den første tredjedelen har ganske jevn intensitet og det samme kan vi se fra wavelet analysen i *Figur 32.* I wavelet analysen kan man i tillegg se at den variasjonen som er i den første tredjedelen kommer fra høyere frekvenser enn hoveddelen av signalet. Når det er variasjon i intensiteten til signalet kan vi også se at variasjonen for det meste skjer på de høyere frekvensene også innad i tonene (sagform).

I fourier transformen ser det ut som om det er en frekvens vi har mye av, og at det er omtrent like mye av frekvensene under og over. Dette kan vi se grunnen til på wavelet analysen. Det er egentlig bare to toner her, en i midten og en under, men det er en god del overtoner som gjør at det virker som om det også er en over. En annen ting jeg ikke klarte å høre på lydfilen var at den andre lavere tonen egentlig består av omtrent like mye av den lavere tonen og den over. Og den høyere tonen har alltid en liten del av den lavere tonen i seg.

Konklusjon

Wavelet analyse er en teknikk som kan brukes for å se på hvordan frekvensen endrer seg med tiden. Det er en teknikk som ligner veldig på fourier analyse, men fourier analyse ser bare på frekvensene som forekommer i løpet av hele signalet – det har ingen tidsperspektiv. Wavelet analyse er veldig nyttig når frekvensene i signalet forandrer seg hyppig. En wavelet er avhengig av to parametere, skala og ω. Skalaen er en størrelse som blant annet kan erstattes med frekvens. ω er et parameter som bestemmer frekvensoppløsningen og tidsoppløsningen.

Kontinuerlig wavelet transform er når en wavelet multipliseres med et signal for så å summere alle produktene for hvert tidspunkt. Dette gjør at man får et kontinuerlig vindu som ved hjelp av forskjellige skalaer får med seg mange frekvenser, og vi får som resultat hvilke frekvenser vi har ved hvert tidspunkt. Denne transformen er programmert på to måter. En intuitiv måte der alt er gjort rett fram og en mer effektiv måte som benytter seg av et triks at den fourier transformerte av to funksjoner ganget med hverandre er den samme som produktet mellom de fourier transformerte av de to funksjonene.

Analysen viser oss at det er mange detaljer vi kan finne ved hjelp av wavelet transform som ikke er mulig å få ut av fourier transform og tidsbildet. Disse detaljene går i hovedsak ut på at vi kan ha flere frekvenser samtidig eller at frekvensene endrer seg med tid. Wavelets gir uansett et fint grafisk bilde av både frekvensutviklingen og tidsbildet i et og samme plot.

Vedlegg 1 - Dataprogrammet brukt i Deloppgave 1

```
% FYS2130 - Prosjekt
% Deloppgave 1e

clear all;
clear figure;

dt = 0.1; % tidsintervall
N = 256; % Antall punkter langs x-aksen
s = 32*dt; % Skala
w = 6; % omega, vinkelfrekvens
n2 = N/2; % Waveletens midtpunkt

% for w = 2:10
% for t = 2:12
%     s = t*dt;

    psi = zeros(1,N);
    psi = psi + i.*psi;
    omhyllingskurve = zeros(1,N);

    integral_of_psi_real = 0;
    integral_of_psi_imag = 0;

    for n = 1:N
        eta = (n2-n)*dt/s;
        psi(n) = pi^(-1/4) * sqrt(dt/s) * exp(i*w*eta) * exp(-(eta^2)/2);

        integral_of_psi_real = real(psi(n)) + integral_of_psi_real;
        integral_of_psi_imag = imag(psi(n)) + integral_of_psi_imag;

        omhyllingskurve(n) = pi^(-1/4) * sqrt(dt/s)*exp(-(eta^2)/2);
    end

    % Fouriertransformering av waveleten
    F_psi = fft(psi);

    % Plotter realdelen av waveleten
    figure(1);
    plot(real(psi));
    title('Morlet Wavelet - Realdel')
    xlabel('tidssteg')
    ylabel('Wavelet')
    hold on;

    % Plotter kvadratet av absoluttverdien av waveletens realdel
    plot((abs(real(psi))).^2,'r');
    plot(omhyllingskurve,'g');
    legend('Wavelet','Omhyllingskurve')
    hold off;

    % Plotter imaginærdelen av waveleten
    figure(2);
    plot(imag(psi));
    title('Morlet Wavelet - Kompleksdel')
    xlabel('tidssteg')
    ylabel('Wavelet')
    hold on;
```

```
    % Plotter kvadratet av absoluttverdien av waveletens imaginærdel
    plot((abs(imag(psi))).^2,'r');
    plot(omhyllingskurve,'g');
    hold off;

    % Plotter kvadratet av absoluttverdien av den fourier transformerte av
waveleten
    figure(3)
    plot(abs(F_psi).^2);
    title('Morlet Wavelet - Fouriertransformert')
    xlabel('f')
    ylabel('|f|^2')

    % Skriver ut integralene
    fprintf('\n\nScale = %20.10f\nReal integral = %8.20f\nImag integral =
%8.20f', s, integral_of_psi_real, integral_of_psi_imag);
% end
% end
```

Vedlegg 2 - Dataprogrammet brukt i Deloppgave 2

```
% FYS2130 - Prosjekt
% Deloppgave 2

tic; % Tar tiden på programmet

clf;
clear all;
clc;

% Read wav file
N = 2^15; % Hvor stor del av lydfilen som skal tas med
nstart = 1; % Her kan man bestemme starttidspunktet for lydfilen
nstop = nstart + N - 1;

[y, Fs, nbits] = wavread('M:\fys2130\Granmeis2.wav', [nstart nstop]); %
Windows

dt = 1/Fs;
t = (0:N-1)*dt; % En tidsarray med alle tidsbitene fra tid 0 til tid
(N-1)*dt.

% Plotter det opprinnelige signalet
figure(10);
plot(Fs*t,y)
title('Opprinnelig signal')
xlabel('tid (millisekunder)')

% Spiller av lyden
%sound (y, Fs, nbits);

w = 6; % Perioder innenfor omhyllingskurven

% Scaling
s0 = 5*dt; % Første skalaverdi
s1 = 21*dt; % Siste skalaverdi
sN = 20; % Antall skalaverdier vi vil bruke
scaling_vector = linspace(s0,s1,sN); % Lager skalaverdiene, (første s,
siste s, antall s);
s = scaling_vector;
freq = w./(2*pi*s);

wl_sum = zeros(sN,N);
wl_sum = wl_sum + i.*wl_sum;

for s2 = 1:sN
    % Nå trenger jeg en matrise med midtpunktet, og en matrise med
    % wavelet-verdiene.

    % Beregne waveleten
    wlN = 160; % Waveletens utstrekning (med god margin for å klare skala
"21")
    psi = zeros(1,wlN); % Initierer waveleten psi
    psi = psi + i.*psi; % Definerer psi som kompleks
    wlN_mid = wlN/2; % Waveletens midtpunkt

    for n = 1:wlN
        eta = (wlN_mid - n)*dt/s(s2);
```

```
        psi(n) = pi^(-1/4) * sqrt(dt/s(s2)) * exp(i*w*eta) * exp(-
(eta^2)/2);
    end

    % Fordi waveleten starter og slutter utenfor dataene legger jeg til
    % 0-verdier både foran og bak på dataene. Fordi produktet blir null
    % blir summen null og dataene blir ikke påvirket annet enn at de blir
    % dempet i kantene (som de ville blitt uansett).

    y_pad = padarray(y,[(wlN_mid-1) 0]); % Legger til wlN_mid-1 nuller før
og etter

    % Multiplisere waveleten med dataene (y) og summe alt opp, waveletens
    % første element skal ganges med dataenes første element, helt til
    % waveletens siste element ganges med dataenes siste element.

    % For å ikke regne ut dette hver gang løkken kjøres
    start_m = (wlN_mid);
    stop_m = length(y_pad)-(wlN_mid-1);
    start_n = -(wlN_mid-1);
    stop_n = (wlN_mid);

    for (m = start_m:stop_m)
        o = 0;
        for (n = start_n:stop_n)
            o = o + 1;
            wl_sum(s2,m+start_n) = y_pad(m+n) * psi(o) + wl_sum(s2,m
+start_n);
        end

    end

end

figure(1);
plot(real(psi));
title('Morlet Wavelet - Realdel')
xlabel('tidssteg')
ylabel('Wavelet')

figure(3);
imagesc(t,freq,real(abs(wl_sum)));
title('Wavelet Transform')
xlabel('Tid [s]')
ylabel('Frekvens [Hz]')
set(gca,'YDir','normal') % Setter frekvensen øverst

% Skriver ut tiden programmet brukte
script_time = toc;
fprintf('\n\nKjøringstid = %15.5f\n\n', script_time);
```

Vedlegg 3 - Dataprogrammet brukt i Deloppgave 3

```
% FYS2130 - Prosjekt
% Deloppgave 2

tic; % Tar tiden på programmet

clf;
clear all;
clc;

% Read wav file
N = 2^15; % Hvor stor del av lydfilen som skal tas med
nstart = 1; % Her kan man bestemme starttidspunktet for lydfilen
nstop = nstart + N - 1;

[y, Fs, nbits] = wavread('M:\fys2130\Granmeis2.wav', [nstart nstop]); %
Windows

dt = 1/Fs;
t = (0:N-1)*dt; % En tidsarray med alle tidsbitene fra tid 0 til tid
(N-1)*dt.

data_time = (N-1)*dt; % lengden på signalet i sekunder

% Plotter det opprinnelige signalet
figure(10);
plot(Fs*t,y(1:N))
title('Opprinnelig signal')
xlabel('tid (millisekunder)')

% Spiller av lyden
%sound (y, Fs, nbits);

F_y = fft(y,N); % Fourier transform av y, kommer til å bruke F_y(1:N) fordi
lyden har to kanaler
frekv_fft = (Fs/2)*linspace(0,1,N); % Frekvensvektor til fourier transform

% Plotter den fourier transformerte av signalet
figure(1);
plot(frekv_fft,2*abs(F_y(1:N)))
title('Absolutt-verdier av frekvensspekteret')
xlabel('Frekvens (Hz)')
ylabel('|F(frekv)|');

w = 6; % omega0, ca antall bølgetopper i en wavelet

s0 = 5*dt; % Initial scale
s1 = 20*dt; % Final scale, bare for lineær
dds = 0.10; % oppløsningsfaktor, for logaritmisk skala
sN = fix(8/dds) % 1:dds intervaller per oktav, 11 oktaver

% Velg logaritmisk eller lineær skala. 1 for lineær, 2 for logaritmisk
s_choice = 2;
if (s_choice == 2)
    s = s0*2.^((0:sN-1)*dds); % logaritmisk skala
else
    s = linspace(s0,s1,sN) % lineær skala
end
```

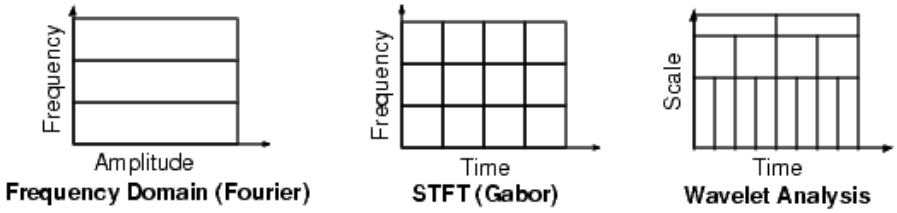
```
sf = s./dt; % Skalafaktor

freq = w./(2*pi*s);

wlN = 160; % Waveletens utstrekning (med god margin for å klare skala "21")
F_psi = zeros(sN,N);

F_both = zeros(sN,N);
f_both = zeros(sN,N);

for s2 = 1:sN
    % Ingen fourier transform funksjon her, for dette uttrykket er allerede
    % fourier transformert
    F_psi(s2,1) = 0;
    for n = 2:(N/2+1)
        argument(n) = -0.5*((sf(s2)*2*pi*(n-1)/N) - w).^2; % (dvs
elementvis kvadrert)
        F_psi(s2,n) = sqrt(2.0*sf(s2)*pi^(+0.25))* exp(argument(n));
    end

    % Multiplikasjon av F_psi og F_y
    F_both(s2,:) = F_psi(s2,:).*F_y(1:N);

    % Invers transform
    f_both(s2,:) = ifft(F_both(s2,:));
end

plot_fboth = sqrt(abs(f_both)); % Definerer hvilke funksjoner som skal
virke på f_both

figure(2);
title('...')
xlabel('Tid [s]')
ylabel('Frekvens [Hz]')

imagesc(t,freq,plot_fboth);
set(gca,'YDir','normal') % Setter frekvensen øverst

if (s_choice == 2) % Hvis logaritmisk skala så fjerner jeg y-aksen
   set(gca, 'YTickLabelMode', 'Manual')
   set(gca, 'YTick', [])
end

hold on

% Cone of Influence (COI)
approx = 1000; % Waveletens utstrekning slutter når verdien av den er
1/approx av maks
wl_width = ((s/dt) * 2*sqrt(2*log(approx)));
wl_width_vector = zeros(sN,N);
max_fboth = max(max(plot_fboth));

% COI shading all values outside
for s2 = 1:sN
    for n = 1:3:fix(wl_width(s2)/2)
        wl_width_vector(s2,n) = max_fboth - plot_fboth(s2,n);
        wl_width_vector(s2,N-n) = max_fboth - plot_fboth(s2,N-n);
    end
end

% Plotter de opprinnelige verdiene pluss coi
```

```
plot_coi = wl_width_vector+plot_fboth;
imagesc(t,freq,plot_coi)
colorbar

hold off

% Skriver ut tiden programmet brukte
script_time = toc;
fprintf('\n\nKjøringstid = %15.5f\n\n', script_time);
```

Vedlegg 4 – Forklar kontinuerlig wavelet transform til en medstudent (Deloppgave 5)

Wavelet-analyse er en teknikk lignende fourier-analyse, men med forskjellige parametere. Fourier-analyse gjør om et signal fra å se på tidsutviklingen av det til å se på hvilke frekvenser det inneholder, mens wavelet-analyse er et 3D-plot der man ser på både tidsutviklingen og hvilke frekvenser hvert tidspunkt inneholder. Fordi fourier-transformen ikke inneholder et tidsperspektiv mister vi mye informasjon hvis signalet forandrer seg over tid. Fourier-transform er best når vi ser på et repeterende signal. Målet med denne teksten er å forstå hvorfor wavelet-analyse er et godt alternativ til fourier-analyse.

Man kan tenke seg at det også er mulig å se på tidsutviklingen ved hjelp av en teknikk kalt stykkevis fourier-transform, der man deler opp et signal i mange små biter, kalt vinduer, og gjør fourier-transform på hvert vindu. Dette er en nyttig teknikk, men problemet med den er at oppløsningen på vinduene er konstant og vi vil kunne miste informasjon i signaler der frekvensen endres mye. Dette kan forbedres ved å la vinduene overlappe hverandre for så å sette sammen informasjonen etterpå.



Figur 1. Fourier-transform gir amplitude for hver frekvens, stykkevis fourier (STFT) gir frekvenser (og deres amplitude i et 3D-plot) med faste tidsintervaller. Wavelet-analyse gir en skala, som kan konverteres til frekvens,på samme måte som STFT, men man kan samtidig forandre på tidsintervallet for hvert vindu.

Kontinuerlig wavelet transform er basert på at vinduene overlapper, og de blir flyttet kontinuerlig ett steg om gangen fra den ene enden til den andre. Dette kjøres flere ganger med forskjellige størrelser slik at man er sikker på at man får med seg hele frekvensspekteret. Det er også mulig å forandre på steglengden slik at det ikke er en kontinuerlig forflytning, dette kan gå mye fortere kalles for diskret wavelet transform, men skal ikke gjøres her. I kontinuerlig wavelet transform kan man selv velge størrelsene på vinduene helt fritt, men det kan man ikke på den diskret måten – nok en grunn for at det heter kontinuerlig.

Wavelet transform ligner ganske mye på fourier transform. Der fourier transform er basert på sinus og cosinusbølger med varierende frekvens er wavelet transform basert på små bølger, kalt wavelets, med varierende bredde og posisjon. Fourier transform er gitt ved ligningen til venstre og wavelet transform er gitt ved ligningen til høyre

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \qquad \gamma(s,\tau) = \int_{-\infty}^{\infty} f(t) \Psi^*_{s,\tau}(t) dt$$

der f(t) er signalet som transformeres, exp(-iωt) er sinus og cosinusbølgene brukt i fourier transform og $\Psi^*_{s,\tau}(t)$ er den kompleks konjugerte av waveleten.

Waveletens bredde og posisjon varieres. Disse er gitt ved variablene s og τ. Bredden, s, kalles skalaen til waveleten og er et mål på frekvensen. Denne virker på samme måte for en wavelet som for en vanlig sinusfunksjon. Hvis man har en sinusfunksjon sin(t/s) vil frekvensen til funksjonen øke proporsjonalt med s, og funksjonen blir mer sammentrykt. På samme måte vil en wavelet bli mer sammentrykt når s øker. Når waveleten er liten vil den være følsom for høye frekvenser i signalet, mens en større wavelet vil være bedre egnet for lave frekvenser. τ er en variabel med nær sammenheng til tid, men som viser forflytningen til waveleten når waveleten flyttes bortover signalet som et vindu. Forflytning for en sinusbølge er gitt ved sin(t – n) der n er forflytningen, og det er på akkurat samme måte for wavelets som sett i ligningene nedenfor. En wavelet kan ha veldig mange forskjellige former. Dette er en av fordelene med wavelet-analyse fordi waveletene kan tilpasses de dataene man har om man er flink.