0.1 Morlet wavelets i FYS2130-notasjon (v 28.04.11)

I wavelet-formalismen opererer vi ofte med en moder-wavelet som trekkes ut ved hjelp av en skaleringsfaktor for å lage såkalt wavelet-døtre. Dette er en betraktningsmåte som egner seg godt, generelt sett, fordi en wavelet kan ha svært mange ulike former, alt etter hva slags signal vi ønsker å analysere. Dessuten er skalerings-begrepet ganske naturlig å bruke når vi arbeider med diskret wavelet-transformasjon.

I vår sammenheng kommer vi kun til å bruke Morlet wavelets, og bare i såkalt kontinuerlig wavelettransformasjon. I en slik sammenheng bestemmer vi intensiteten for ulike frekvenser ved ulike tider i et signal. Det sentrale begrepet er da *frekvens*, det vil si den frekvensen vi velger å analysere signalet med hensyn på. Vi kan gjerne også i en slik sammenheng bruke "skalering" som parameter, men denne variablen er ikke like naturlig å bruke fordi vi da må regne om for å se hva en skaleringsfaktor svarer til med hensyn til frekvens. Også når det gjelder andre størrelser vil det i vår sammenheng være enklere å bruke en spesiell utformet formalisme i stedet for den vanlige wavelet-formalismen.

Det finnes forøvrig neppe noe som kan kalles "vanlig waveletformalisme". Ulike forfattere skriver sine uttrykk på forskjellig vis. Uttrykkene kan se ganske enkle og greie ut ved første blikk, men når vi sjekker detaljene, er det gjerne slik at en t i formalismen slett ikke betyr tid i vanlig forstand, men en mer eller mindre komplisert parameter hvor tid inngår. På samme måte er en ω som oftest slett ikke en vinkelfrekvens, men en parameter som i beste fall er direkte relatert til vinkelfrekvens.

Det kan være nyttig i denne sammenheng å huske at Morlet wavelets først ble introdusert i 1984. Wavelets er derfor en meget ung formalisme i matematisk målestokk. Når det atpåtil er slik at wavelet-analyse brukes på svært mange ulike måter, er det ikke rart at formalismen ikke ennå har "satt seg".

Vi vil nå forsøke å gi formalismen for Morlet-wavelets på en ny måte i et håp om at denne skrivemåten vil gjøre det betydelig lettere å arbeide med kontinuerlig wavelettransformasjon der frekvensinnhold er det sentrale.

En Morlet wavelet vil i denne formalismen skrives på formen:

$$\Psi_{(f_a,t',K)}(t) = C_1 e^{i2\pi f_a(t-t')} e^{-\frac{1}{2}\left(\frac{2\pi f_a(t-t')}{K}\right)^2} \tag{1}$$

Her er f_a frekvensen vi ønsker å analysere signalet for (med akkurat denne waveleten). Morlet-waveleten beskrives som en funksjon av tiden t, og midtpunktet for waveleten vil finnes ved tiden t'. K er en "bølgetalls-parameter". Vi kommer tilbake til denne om litt.

Konstanten C_1 kan gis på mange ulike måter. Vi har valgt en variant som gir en pen

symmetri med en annen konstant som vi kommer tilbake til, og da er uttrykket som følger:

$$C_1 = \sqrt{\frac{\sqrt{2\pi} f_a}{K f_s}} \tag{2}$$

Her er f_a lik frekvensen vi ønsker å analysere signalet med hensyn på. Parameteren f_s er samplingsfrekvensen når vi gjør konkrete beregninger på et konkret signal, og signalet er samplet med denne frekvensen. Waveleten må da beskrives i samme tidsoppløsning som det samplede signalet.

Merk at Morlet-waveleten er kompleks. Uttrykket $e^{i\omega_a(t-t')}$ består jo av et reelt cosinus-ledd såvel som et imaginært sinusledd. Dersom vi ønsker å plotte Morleten, må vi plotte realdel og imaginærdel hver for seg ved å bruke $real()$ og $imag()$ dersom vi bruker Matlab (eller tilsvarende funksjoner i andre programmeringsspråk). De to diagrammene kan med fordel plottes i samme diagram, men med forskjellig farge.

0.1.1 Kontinuerlig wavelet transform, basisligning

Et wavelet-diagram (også kalt skalogram eller sonogram) er en grafisk fremstilling av innholdet i en todimensjonal array. Langs x-aksen har vi tid, langs y-aksen frekvens. I prinsippet beregner vi hvert enkelt arrayelement på følgende måte:

$$W_K(t', f_a) = \Sigma_{t=0}^{T} g(t) \Psi_{(f_a, t', K)}(t) \tag{3}$$

hvor $g(t)$ er det signalet vi ønsker å analysere. Dersom denne tidsarrayen består av N punkter, og vi beskriver waveleten i like mange punkter, betyr det at vi må gjennomføre N multiplikasjoner for hvert eneste punkt i arrayen. Dersom vi flytter waveleten med ett hakk i tiden for hver ny beregning, betyr det at vi må gjennomføre totalt N^2 multiplikasjoner og N^2 summasjoner for hver eneste frekvens f_a vi ønsker å analysere for. Dersom vi f.eks. ønsker å analysere for 100 ulike frekvenser, betyr det at det inngår $N^2 \cdot 100$ ($\approx 4.2 \cdot 10^{11}$ dersom $N == 2^{16} = 65536$) multiplikasjoner og addisjoner for å få hele wavelet-diagrammet. Det ville kreve en betydelig regnekapasitet.

0.1.2 Kontinuerlig wavelet transform, mer effektiv algoritme

Det er en smartere måte å gjøre dette på. Hovedideen er basert på "digital spektral filtrering" ved hjelp av fouriertransformasjoner. Svært enkelt fortalt går metoden ut på at vi først fourieromvender tidssignalet vi skal analysere. Det fourieromvendte signalet er en kompleks array der første arrayelement gir gjennomsnittsverdien, andre arrayelement gir amplituden for den frekvensen som har nøyaktig en bølgelengde innenfor tidsstrengen og så videre (slik vi kjenner fra kapittel 3). Dersom vi nuller ut alle elementene bortsett fra

ett element (komplekst), og foretar en invers fouriertransformasjon, får vi en sinusfunksjon (med korrekt fase) som vil strekke seg over hele tiden tidssignalet ble samplet.

Selv om signalet hadde den aktuelle frekvensen bare en kort stund en eller annen gang i løpet av tiden T, vil vi altså ved denne ekstremt strenge filtreringen (utvelgelse av bare ett element i frekvensspekteret) få et tidssignal med identisk amplitude i hele T.

Dersom vi foretar en mindre streng filtrering, og beholder frekvenskomponentene i et helt område langs frekvensaksen, vil vi etter tilbaketransformering få et tidsoppløst signal. Dersom frekvensen f_1 bare forekom en kort stund mellom tiden t_1 og t_2, vil vi etter tilbaketransformeringen få fram nettopp at signalet med denne frekvensen bare gjenfinnes i det aktuelle tidsintervallet (pluss litt til, avhengig av hvor bredt område vi beholdt ved den digitale filtreringen). Alt annet i det opprinnelige signalet vil da være fjernet i det tilbaketransformerte signalet.

Hittil har vi omtalt digital frekvensfiltrering i fourierspekteret slik at vi har beholdt korrekte verdier i et helt lite intervall av frekvensspekter-arrayen, og nullet ut arrayen på begge sider av dette intervallet.

Utvelgelsen av hvilke frekvenskomponenter som skal beholdes kan gjøres litt mer gradvis, ved at vi multipliserer amplitudene i frekvensspekteret med en klokkefunksjon, f.eks. en gaussisk funksjon, med toppunktet i den frekvensen vi er mest interessert i. Klokkefunksjonens bredde må finnes ved litt prøving og feiling.

Det er nettopp dette vi gjør i den effektive varianten av wavelettransformasjon. Det kan vises matematisk at dersom vi multipliserer den komplekse fouriertransformerte av tidsfunksjonen vår element for element med den fouriertransformerte til waveleten, og tilbaketransformerer med en invers fouriertransformasjon, får vi akkurat samme resultat som om vi hadde gjennomført den opprinnelige transformasjonen i tidsbildet alene (likning (3)). Det elegante med prosedyren med å gå via fouriertransformasjon og omvendt fouriertransformasjon er at vi får beregnet wavelettransformasjonen i alle tidspunkt samtidig for den analysefrekvensen vi har valgt.

Det kan være interessant å se hvor mye vi sparer i tid på en slik effektiv algoritme. Fourieromvendingen av det opprinnelige tidssignalet g behøver vi bare å gjøre én gang for hele wavelet-diagrammet. Den fourieromvendte til g betegner vi med $\hat{g}$. For hver frekvens må vi beregne den fouriertransformerte $\hat{\Psi}$ av waveleten. Denne kan vi beregne direkte ut fra følgende formel:

$$\hat{\Psi} = C_2 e^{-\frac{1}{2}(K-\eta]^2} \tag{4}$$

hvor $C_2 = 1/C_1$ der

$$\eta_n = \frac{K f_s}{N f_a} \cdot n \tag{5}$$

er det n-te elementet i arrayen η. $(n = 1, 2, ..., N)$

Med normalt smart programmering er det N multiplikasjoner som trengs for å lage $\hat{\Psi}$. Dernest må vi multiplisere $\hat{\Psi}$ elementvis med $\hat{g}$. Det gir nye N multiplikasjoner. Dernest skal vi foreta en invers fouriertransformasjon av produktet. Tiden som går med til en invers fast fourier transformasjon er proporsjonal med N multiplikasjoner i motsetning til en vanlig fouriertransformasjon som krever N^2 multiplikasjoner (gjennomført omtrent etter samme mønster som i ligning (3)).

Etter den inverse transformasjonen får vi i vårt tilfelle generelt et signal som er komplekst. Vi velger av ulike grunner å betrakte absoluttverdien av resultatet. Beregningen av absoluttverdien krever nye $2N$ kvadreringer og N rotutregninger.

Alt i alt går det altså med i størrelsesorden (meget grovt) 5-6 x N multiplikasjoner for hver linje i det endelige wavelet-diagrammet med denne fremgangsmåten. Dette er betydelig mindre enn de ca N^2 multiplikasjonene som ville medgå ved bruk av algoritmen gitt i ligning (3).

Algoritmen må så gjentas for så mange analysefrekvenser som vi ønsker å bruke. Antar vi at vi ønsker å bruke 100 analysefrekvenser, betyr det at det medgår i størrelsesorden 500-600 x N multiplikasjoner. For $N = 2^{16} = 65536$ vil dette si ca $3.9 \cdot 10^7$ multiplikasjoner mot $4.2 \cdot 10^{11}$ for den opprinnelige algoritmen. Det er en besparelse på ca $N/6 \approx 10000$ ganger, med andre ord en meget stor forbedring for store datastrenger.

0.1.3 Valg av frekvenser i analysen

Ved en vanlig fast fourier transform (FFT) vil frekvensspekteret inneholde frekvenser fra og med 0 til samplingsfrekvensen. Avstanden mellom en frekvens og den neste er $1/T$, det vil si at frekvensskalaen i en FFT er lineær.

I en del ulike sammenhenger kan det imidlertid være en fordel å heller velge analysefrekvensene i en logaritmisk skala. Med det mener vi at dersom vi f.eks. starter med frekvensen $f_{a,0}$ og skal finne den neste, får vi den ved hjelp av en multiplikasjon med et fast frekvensforholdstall dd slik:

$$f_{a,1} = f_{a,0} \cdot dd$$

...

$$f_{a,j+1} = f_{a,j} \cdot dd$$

...

Ved en kontinuerlig wavelettransformasjon kan vi selv velge hvilket frekvensintervall vi ønsker å analysere og hvor mange frekvenser som skal analyseres innenfor dette intervallet. Det kan være nyttig å foreta en vanlig fouriertransformasjon av tidssignalet først for å se hvilke frekvensområde det synes å være noe interessant i. Anta at vi ut fra fourierspekteret

ønsker å analysere for frekvenser i intervallet $[f_{min}, f_{max}]$. Anta at vi videre ønsker å ha M linjer (M ulike analysefrekvenser) i analysen. I så fall må vi velge:

$$dd^{M-1} = \frac{f_{max}}{f_{min}} \tag{6}$$

Følgelig må

$$dd = \left(\frac{f_{max}}{f_{min}}\right)^{1/(M-1)} \tag{7}$$

0.1.4 Optimalisering

Plotter vi ligning (4) for en og samme f_a, men ulike K, vil vi se at bredden på den gaussformede kurven $\hat{\Psi}$ er omvendt proporsjonal med K. Desto flere perioder vi får plass til innenfor omhyllingskurven i waveleten, desto mer presist får vi definert frekvenser i analysen.

Vi får imidlertid ikke både i pose og sekk. En større K vil i vår sammmenheng si at hele waveleten vil strekke seg over et større tidsintervall enn for en mindre K. Dersom vi så har den opprinnelige prosedyren gitt i ligning (3) i bakhodet, forstår vi at en skarp detalj i tiden vil bli smurt mer utover i tid ved en stor K enn for en liten K.

Med andre ord er det slik at ved å velge en liten K får vi i prinsippet en god tidsdefinisjon av detaljer i signalet, men en dårlig frekvensdefinisjon, mens for en stor K blir det akkurat motsatt.

Det kunne tenkes at det da ville være lurt å foreta to wavelettransformasjoner, en med så liten K som mulig (det anbefales ikke å la K være mindre enn 6, ellers blir det mye utilsiktet feil som sniker seg inn), pluss en wavelettransformasjon med så stor K som mulig. Disse to skulle man da tro ville gi et perfekt totalbilde.

Slik er det ikke i praksis. Signalet glir gjerne fra en frekvens til en annen i løpet av tiden. Både meget høy og lav K-verdi vil forkludre waveletbildet så mye at resultatet ville ha liten verdi.

Vi må derfor prøve oss fram med ulike K-verdier for å finne en optimal løsning. Denne optimale løsningen er bestemt av signalet selv. Ethvert signal har sin egen effektive "tidskonstant" og sin egen "frekvensfordeling". Husk for eksempel hvordan sammenhengene var ved resonans i kapittel 1. Systemer med en høy Q-verdi responderte bare på et lite frekvensområde (hadde en smal frekvensresponskurve), men brukte da lang tid på å komme i gang med oscillasjonene og lang tid på at oscillasjonene døde ut etter at den ytre kraften ble slått av (det vil si lang tidskonstant). Velger vi en passe K som svarer til signalets egne karakteristika, får vi et wavelet-diagram med nær optimalt tids- og frekvens-oppløsning samtidig!

Enda en form for optimalisering er viktig for å få et bra wavelet-diagram. Brukes f.eks. funksjonen *imagesc( )* i Matlab for å plotte wavelet-diagrammet, vil vi få en fargekoding der rødt svarer til den maksimale verdien i wavelet-diagram-arrayen. Punkter med halvparten av denne verdien vil bli plottet som grønt, og punkter med f.eks. 10 prosent av maksimal verdi vil få en farge nær blåfargen som angir minste verdi.

Dersom vi er interessert å se detaljer også i signaler som er ganske svake, må vi forsterke nedre del sammenlignet med maksimum. Det kan vi oppnå ved å f.eks. ta kvadratroten til alle verdiene i wavelet-diagram-arrayen før resultatet plottes.

Omvendt, dersom vi ønsker å undertrykke de svake signalene og heller fokusere på små relative endringer i nivåene nær opp til de maksimale, kan vi f.eks. undertrykke de svake signalene ved å ta kvadratet av wavelet-diagram-arrayen.

Vi bør med andre ord forsøke oss fram til hva som passer best for den type analyse vi ønsker å gjennomføre.

0.1.5 Oppsummering

Som oppsummering kan vi si at en kontinuerlig wavelet transformasjon er en meget slagkraftig metode for å analysere tidsavhengige signaler med ulike frekvenser. Metoden krever imidlertid litt teft fra brukerens side for at resultatet skal bli best mulig. Det er fire ulike valg som må vurderes:

- Frekvensområde for analysen. Ofte er det nyttig å ta utgangspunkt i et vanlig fourierspekter av signalet for å velge frekvensområdet.
- Hvor mange ulike frekvenser skal vi bruke i analysen innenfor det aktuelle frekvensområdet nevnt i punkt 1.
- Vi må velge antall bølger innenfor omhyllingskurven i waveleten, dvs vi må velge K i formalismen ovenfor.
- Vi må velge en egnet transformasjon av de endelige dataene i wavelet-arrayen, for at fargenyansene i det endelige plottet skal gi mest mulig informasjon av interesse.

I tillegg er det selvfølgelig også en annen utfordring, nemlig å plukke ut et interessant tids område i en ofte ganske lang datafil. Vi må også velge lengde på datastrengen vi ønsker å analysere. Generelt sett er det lurt å arbeide med relativt korte datastrenger mens uttesting av parametre foregår, og så bruke lengre strenger i de endelige analysene (dersom det er ønskelig). Siden en invers fast fourier transform inngår i beregningene, bør lengden på datastrengen være en toer-potens, f.eks. $2^{15} = 32768$ punkter.

På min laptop går det greit med opp til $2^{17} = 131072$ punkter i datafilene og opp til 200 ulike frekvenser i analysen. Beregningene tar da noen få sekunder.