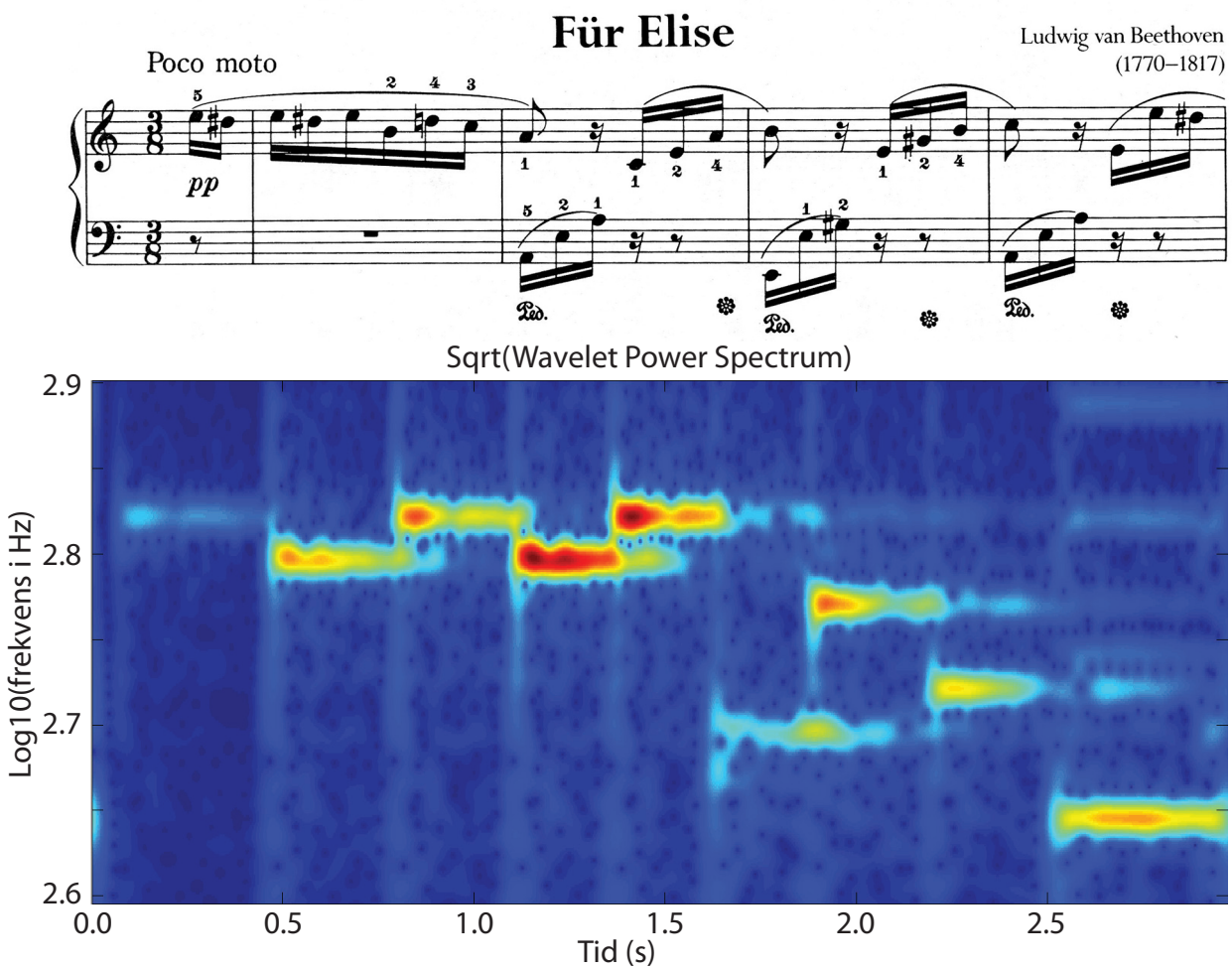
Kapittel 14

Wavelettransformasjon



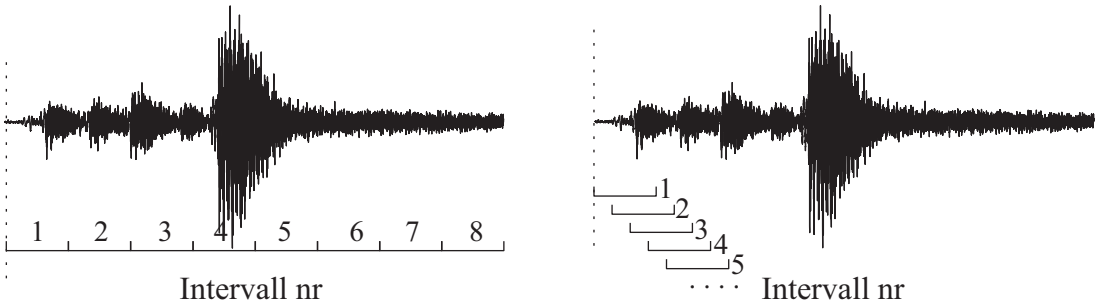
Ønsker du deg en samtidig frekvens- og tidsanalyse, er wavelettransformasjon en spennende metode. Moderne wavelettransformasjon har ofte store fordeler framfor klassisk fouriertransformasjon. Atpåtil er det utfordrende analogier mellom waveletanalyse og Heisenbergs uskarphetsrelasjon. Håper ditt møte med wavelets blir en spennende reise inn i nytt terreng!

14.1 Hva slags info kan wavelettransformasjon gi oss?

Anta at vi synger en "a" og holder tonen hele tiden mens vi digitaliserer lyden. Fourieromvendes signalet vil vi se et frekvensspekter med grunntone og overharmoniske temmelig likt det vi finner fra analysen av en ren tone fra et musikkinstrument. For "stasjonære" signaler (endrer ikke nevneverdig karakter underveis) er fouriertransformasjon en fantastisk metode for å hente ut verdifull informasjon.

Anta derimot at vi digitaliserer skarpe trommelyder med nesten helt stille partier innimellom og foretar fouriertransformasjon. Da vil vi se at fouriertransformasjon egentlig er en dårlig metode ved analyse av signaler som endrer karakter med tiden. Når vi tar ut absouluttverdien fra en fouriertransformasjon, forsvinner enhver tidsinformasjon, mens frekvensoppløsningen er god. Men hva betyr egentlig frekvens i et signal som endrer karakter så mye underveis? Det spørsmålet bør vi tenke litt over, slik at vi ikke bruker en slagkraftig metode på et datamateriale der metoden egentlig ikke egner seg!

Vi kan bøte på dette problemet noe ved å stykke opp signalet slik at vi har et noenlunde ensartet signal innenfor hver bit vi analyserer. I så fall anvender vi en metode som kalles stykkevis fouriertransformasjon (se figur 14.1). Den formen for waveletanalyse vi skal diskutere i dette kapitlet ligner litt på stykkevis fouriertransformasjon, men har en rekke fordeler framfor denne. Vi skal i dette kapitlet gi en innføring i waveletanalyse og vise hvordan metoden kan optimaliseres.



Figur 14.1: *For å få tidsinformasjon ved analyse av en lang tidsstreng med data, kan vi stykke opp det totale tidsintervallet og foreta fouriertransformasjon for intervall etter intervall. Intervallene kan velges slik at de ikke overlapper hverandre (venstre del) eller slik at de overlapper hverandre (høyre del). Se teksten for vurdering av disse teknikkene.*

Benyttes stykkevis FT ved at den totale datamengden deles opp i f.eks. m like store biter, skal det noe til at vi får ensartede signaler innenfor hver av bitene. Dersom vi i stedet velger å stykke opp i m biter med varierende lengde slik at det blir et noenlunde ensartet signal innen hver bit, vil det bli store forskjeller i frekvensspekteret fra bit til bit: Analyseresultatet vil bli kritisk avhengig av hvordan vi stykker opp den totale datamengden.

En bedre metode kan kanskje være å velge ett "vindu" (f.eks. 256 datapunkter), og gjøre

analysen først for de første 256 datapunktene i den lange datastrengen vi har, deretter flytte vinduet litt (f.eks. 16 punkter), og gjøre anlysen på punkt 17 til og med punkt 17+255=272, for så å flytte vinduet enda et hakk for neste anlyse.

Ved en slik "glidende filter"-metode unngår vi hopp i resultatene som skyldes tilfeldigheter i hvordan intervallene velges. Ulempen er at vi må gjennomføre til dels mange tilsynelatende unødvendige beregninger. Vi får med andre ord ganske mye overflødige data (engelsk: "redundancy") i resultatene.

Den største ulempen med en glidende stykkevis fouriertransformasjon er likevel at den relative fekvensoppløsningen blir svært forskjellig for høye og lave frekvenser. Vi husker fra et tidligere kapittel at ved en diskret fouriertransformasjon av en tidsstreng med varighet T blir frekvensoppløsningen $1/T$. Anta at vi f.eks. velger $T = 0.5$ s. Da blir frekvensoppløsningen 2 Hz. *Relativ* frekvensoppløsning kan defineres som frekvensoppløsning dividert på signalets frekvens. Ved 50 Hz vil da den relative frekvensoppløsningen bli $2/50 = 4$ %, mens ved et 5 kHz signal vil den relative frekvensoppløsningen bli $2/5000 = 0.04$ %. Vi ser altså at frekvensangivelsen vil bli langt mer nøyaktig for høye frekvenser enn for lave.

"Kontinuerlig waveletanalyse" ligner på en glidende, stykkevis fouriertransformasjon, men gir samme relative frekvensoppløsning for alle frekvensene. Hemmeligheten er å bruke ulik lengde på tidsstrengen alt etter hvilken frekvens vi analyserer.

Resultatet kan fremstilles som et tredimensjonalt plot. Langs x-aksen har vi tid, angitt f.eks. ved midtpunktet i analysevinduet vi bruker. Langs y-aksen har vi frekvensskalaen, og som den tredje komponenten har vi f.eks. intensiteten for bølgen (fourierkomponenten for denne frekvensen kvadrert). Denne informasjonen kan gis i en eller annen form for 3D-graf, f.eks. som "stack-plot", kvotekurver, surface mesh, eller fargekoding for å nevne noen anskuelsesformer.

Det kan nevnes at det også finnes en diskret wavelettransformasjon hvor vi foretar så få transformasjoner som overhodet mulig uten tap av informasjon. En slik transformasjon er mye mer effektiv enn den kontinuerlige, og brukes i teknologiske sammenhenger der det er viktig at ting går fort (for eksempel i MP3-spillere). Ulempen med en diskret wavelettransformasjon er at resultatet er langt vanskeligere å forstå enn et vanlig fourierspekter. Det er hovedgrunnen til at vi ikke går inn på den metoden her.

Waveletanalyse er et omfattende fagfelt innen matematikk/informatikk, og det gis egne kurs om emnet ved mange universiteter. Vi kommer ikke til å gå i detalj om den strengt matematiske eller datatekniske siden av wavelets. Hensikten med å ta med wavelets i denne boka, er å gjøre studenten oppmerksom på at fouriertransformasjon *ikke* egner seg for ikke-stasjonære signaler, og samtidig peke på en analysemetode som ofte er langt å foretrekke i slike sammenhenger. Dessuten kan arbeid med wavelets bidra til en dypere forståelse av tidsavgrensede fenomener og de tilsvarende frekvenser. Blant annet er det nære analogier mellom Heisenbergs uskarphetsrelasjon og waveletanalyse.

En del av dere vil nok bruke waveletanalyse i masteroppgaven eller i et evt. PhD-prosjekt

(og senere jobber). Av den grunn legger vi vekt på å vise hvor waveletanalyse er nyttig og når metoden ikke har mye å gi. Wavelets brukes bl.a. for å analysere solflekkaktivitet (og endringer i syklusen med tiden), El Niño sørlige oscillasjoner i Stillehavet, isbre-sykler, ruhet, kornstørrelseanalyser, analyse av f.eks. kreftceller vs normale celler og mye mer.

Teknologisk er det en omfattende bruk av wavelets bl.a. i jpeg-komprimering av bildefiler, og i mp3-komprimering av lyd.

14.2 Kort historikk

Den franske matematikeren Joseph Fourier (1768-1830) "oppdaget" fouriertransformasjon for vel 200 år siden. (Fourier arbeidet forøvrig med varmestrømning, og var visstnok den første som oppdaget drivhuseffekten.)

Fouriertransformasjon benyttes i stor grad i analytisk matematikk. I tillegg fikk transformasjonen en enorm utbredelse i dataverdenen etter at J.W.Cooley og J.W.Tukey i 1965 oppdaget den såkalte "Fast Fourier Transform" (FFT) som gjør det mulig å foreta en fouriertransformasjon svært mye raskere enn tidligere. Ved FFT benyttes symmetriene i sinus og cosinusfunksjonene for å redusere antall multiplikasjoner ved utregningen, men for å få dette til, må antall datapunkter være en heltalls potens av 2, dvs $N = 2^n$.

Det hevdes at Cooley-Tukeys Fast Fourier Transform egentlig ble oppdaget av Carl Friedrich Gauss ca 1805, men glemt og delvis gjennoppfunnet flere ganger før 1965. Suksessen til Cooley og Tukeys gjenoppdaging skyltes nok at datamaskinen gjorde sitt inntog omtrent på denne tiden.

Waveletanalyse er av langt yngre dato. Riktignok ble wavelets introdusert allerede ca 1909, men metoden ble for alvor først tatt i bruk fra ca 1980 av. Det er langt større spillerom for spesielle varianter av waveletanalyse enn ved fouriertransformasjon. Det er både en fordel og ulempe. Vi kan langt på vei skreddersy en waveletanalyse slik at den passer optimalt for de dataene vi vil analysere. Uelmpen er at den store variasjonsmuligheten medfører at vi må bruke hodet litt mer ved waveletanalyse enn ved fouriertransformasjon, både når transformasjonen skal gjennomføres, og når vi tolker resultatene. Men resultatene blir ofte desto mer interessante!

14.3 Kort om matematikken bak

14.3.1 Oppfrisking av fouriertransformasjon

Vi har gått gjennom fouriertransformasjon i et tidligere kapittel, men la oss repetere de matematiske uttrykkene også her.
La $x(t)$ være en integrerbar funksjon av tid. Da kan vi beregne en ny funksjon $X(\omega)$, hvor ω er vinkelfrekvens, på følgende måte:

$$X(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt \tag{14.1}$$

Det morsomme med denne funksjonen er at vi kan ta en tilsvarende "omvendt" transformasjon:

$$x(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} d\omega \tag{14.2}$$

og ende opp med eksakt den opprinnelige funksjonen igjen. Merk fortegnskiftet i den komplekse eksponensialfunksjonen.

Det finnes tre ulike, vanlige måter å fordele faktoren foran integraltegnet på, vi har valgt den varianten som gir mest symmetriske uttrykk.

Fra likningene (14.1) og (14.2) ser vi at når $x(t)$ er reell, vil $X(\omega)$ være kompleks. Dette er nødvendig for at $X(\omega)$ både skal kunne angi hvor kraftig svingning vi har ved ulike frekvenser, og i tillegg angi innbyrdes fase til de ulike frekvenskomponentene. (En symmetri i X medfører at x etter den omvendte transformasjonen blir reell, slik den var opprinnelig.)

Det bør forøvrig nevnes at x og X generelt sett ikke behøver å være funksjoner av tid og frekvens. Det finnes mange ulike typer funksjoner og sammenhenger hvor fouriertransformasjon anvendes. I vårt sammenheng begrenser vi oss imidlertid (nesten utelukkende) til tid og frekvens.

Uttrykkene ovenfor anvendes ved analytiske beregninger. Når vi bruker datamaskin, kjenner vi ikke fullstendig til hvordan $x(t)$ varierer i tid. Vi kjenner bare verdien av x i diskrete tidspunkt t_n. I disse tidspunktene har x verdiene x_n hvor n er en indeks som varierer fra 1 til N, dersom x er beskrevet i N tidspunkt. Vi antar at det er valgt ekvidistante tidspunkt slik at det er en fast tid mellom to nærliggende tidspunkt. Total tid x er beskrevet over er da $T = N * \delta t$ hvor δt er tiden mellom to tidspunkt i beskrivelsen (detaljer diskutert i et tidligere kapittel).

Når fouriertransformasjon gjennomføres på diskrete data, brukes en diskret transformasjon. Denne kan angis slik:

$$X_k = \frac{1}{N}\sum_{n=1}^{N} x_n e^{-i2\pi f_k t_n} \tag{14.3}$$

hvor $k = 0,\ 1,\ 2,\ \ldots\ ,\ N-1$. Videre er $f_k = 0, f_s/N, 2f_S/N, ..., f_s(N-1)/N$ der f_s er samplingsfrekvensen. Endelig er $t_n = 0, T/N, 2T/N, ..., T(N-1)/N$ der $N/T = f_s$.

Det er kanskje ikke så lett å se ut av uttrykket, men det vi egentlig gjør for å bestemme den fouriertransformerte for en frekvens f_k, er å multiplisere (ledd for ledd) den digitaliserte funksjonen x_n med en cosinusfunksjon med frekvensen f_k og summere alle leddene som da framkommer. (For imagninærdelen av den fouriertransformerte multipliserer vi med en sinusfunksjon med frekvensen f_k.)

Den tilsvarende "omvendte" transformasjonen er da:

$$x_n = \sum_{k=1}^{N} X_k e^{i2\pi f_k t_n} \tag{14.4}$$

for $n = 1,\ 2,\ 3,\ ...,\ N$.

14.3.2 Formalisme ved wavelettransformasjon

Wavlettransformasjon kan angis tilsynelatende på nokså analog måte som en fouriertransformasjon:

La $x(t)$ være en integrerbar funksjon av tid. Da kan vi beregne en ny funksjon $\gamma_K(\omega_a, t)$ som gir informasjon om frekvenser og tid samtidig.
ω_a kan betegnes som "analyse-vinkelfrekvens". K er en "skarphets"-parameter (også kalt "bølgetallet", se siden) knyttet til hvorvidt vi ønsker å ha høy presisjon i tidsangivelser (K liten) eller høy presisjon i frekvensangivelser (K stor).
Enkeltverdier for den nye wavelettransformerte funksjonen kan finnes på følgende måte:

$$\gamma_K(\omega_a, t) = \int_{-\infty}^{\infty} x(t+\tau)\Psi^*_{\omega_a,K}(\tau)d\tau \tag{14.5}$$

Her er $\Psi_{\omega_a,K}(\tau)$ selve waveleten. Asteriksen sier at det er den kompleks konjugerte av uttrykket for waveleten vi må bruke.

Det spesielle med waveletanalyse er at vi kan velge mellom nærmest uendelig mange forskjellige wavelets alt etter hva vi ønsker å få fram i analysen. I vår sammenheng kommer

vi bare til å bruke såkalt Morlet wavelets. Matematisk kan den relle delen av en Morlet wavelet uttrykkes som en *cosinus*funksjon (pluss et bitte lite konstant korreksjonsledd) konvolutert med en gaussisk funksjon ("gaussisk omhyllingskurve"). Den imaginære delen er en *sinus*funksjon konvolutert med samme gaussisk funksjon som i stad.

En Morlet-wavelet kan beskrives som:

$$\Psi_{\omega_a,K}(\tau) = C\{\exp(-i\omega_a\tau) - \exp(-K^2)\} \cdot \exp(-\omega_a^2\tau^2/(2K)^2) \tag{14.6}$$

hvor C er en "normeringskonstant". Når vi beskrver $\Psi_{\omega_a,K}(\tau)$ numerisk, kan vi med fordel bruke følgende uttrykk for C:

$$C = \frac{0.798 \;\; \omega_a}{f_s K} \tag{14.7}$$

hvor f_s er samplingsfrekvensen.

[♠ ⇒ Noen kommentarer:
Det har ikke satt seg en ensartet beskrivelse av wavelets ennå. Forskjellige kilder angir formalismen på ulike måter, der blant annet uttrykk så som "scaling-parameter", "Mother-" og "daughter-wavelets" er sentrale begreper. Vi har valgt å bruke en fremstilling som ligger nær opp til en artikkel av Najmi og Sadowsky (se litteraturlisten) fordi denne ligger temmelig nær opp til øvrig formalisme i denne boka. "Konstanten" C har jeg imidlertid valgt ut fra prøving og feiling ut fra ønsket om at en wavelettransformasjon av et rent sinussignal skal gi amplituden til sinussignalet uansett valg av parametrene ω_a, f_s og K (avvik fra det perfekte er som oftest mindre enn ca 1 %). Det aktuelle uttrykket for en Morlet wavelet kommer vi ikke til å bruke i praksis, bortsett fra et illustrativt eksempel. Ved effektiv wavelettransformasjon kommer vi til å ta utgangspunkt i den fouriertransformerte av waveleten, og beskriver den direkte. Detaljer gis i teksten som følger. ⇐ ♠]

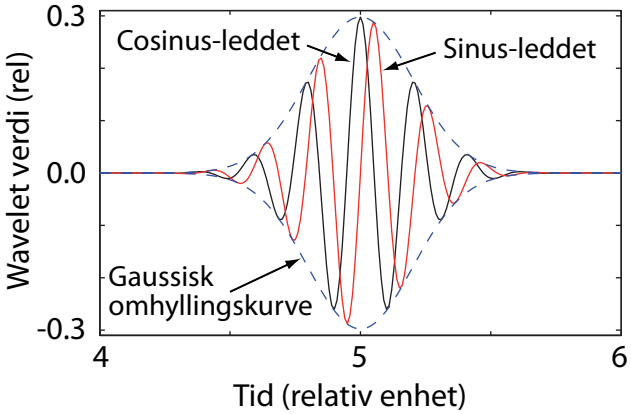
I waveletanalysen sjekker vi om signalet vi studerer inneholder ulike frevenser ved ulike tider. Vinkelfrekvensen vi analyserer for ved en spesifikk wavelet er ω_a. Parameteren τ angir tiden der en spesifikk wavelet har et maksimum, og svarer til senter for det lille tidsintervallet vi undersøker.

Parameteren K er en reell konstant og kan kalles "bredden" på waveleten. Noen kaller den "bølgetallet", fordi den angir omtrentlig antall bølger som det er plass til under den gaussiske omhyllingskurven for waveleten (omhyllingskurven er gitt i siste ledd i ligning (14.6)). Det anbefales at K er 6.0 eller større.

På grunn av det midtre leddet i ligning (14.6) ser vi at waveleten Ψ er kompleks.

Figur 14.2 viser et eksempel på en Morlet wavelet. Vi ser at den bærer navnet med rette: "wavelet" kan nemlig oversettes med "liten bølge".

Vær sikker på at du gjennomskuer hvordan waveleten dannes, nemlig som en gaussisk



Figur 14.2: *Eksempel på en Morlet wavelet for $K = 6$. Både den reelle delen (cosinus-ledd) og den imaginære delen (sinus-ledd) er gitt.*

konvolutering av en kompleks harmonisk funksjon sentrert rundt tiden τ.

Merk at uttrykket i ligning (14.6) er en generell beskrivelse. Når denne skal implementeres i et datamaskinprogram, og analysere et konkret signal, må vi kjenne til samplingsfrekvensen som ble brukt. Denne kommer inn i normeringskonstanten C. Dersom det konkrete signalet er beskrevet i N ekvidistante punkter i tid, er den totale tiden samplingen har foregått lik $T = N/f_s$.

Vi velger da å beskrive enhver Morlet-wavelet vi bruker i analysen ved hjelp av en array med samme samplingsfrekvens og samme lengde på arrayen som det konkrete signalet vi skal analysere.

Dersom vi sammenligner ligning (14.1) med ligning (14.5), ser vi at uttrykkene ligner mye på hverandre. Vi integrerer opp produktet av en funksjon x og en bølge. Begge er derved knyttet til et "indreprodukt" innen matematikken, men som sagt, vi skal bare touche matematikken med en harelabb her.

Det er imidlertid flere forskjeller enn vi først skulle tro. En vesentlig forskjell ligger i at wavelettransformasjonen fører til en tredimensjonal beskrivelse (verdien av γ som funksjon av både ω_a og t), mens en beskrivelse basert på fouriertransformasjon bare er todimensjonal (verdien av X som funksjon av frekvens).

Også for en wavelettransformasjon er det mulig å foreta en "omvendt" transformasjon. Det er essensielt når wavelets brukes i jpeg bildekompresjon og mp3 musikkfil-komprimering. Vi tar imidlertid ikke med detaljer angående denne formalismen i vår sammenheng. (Interesserte henvises til siste referanse i litteraturlisten i slutten av kapitlet.)

14.3.3 "Diskret kontinuerlig" wavelettransformasjon

Først litt om bruk av ordene "diskret" og "kontinuerlig". Et digitalisert signal vil vi kalle diskret, fordi vi bare har et endelig antall måleresultater (ekvidistante i tid). Vi vil likevel betegne den spesielle wavelettransformasjonen som beskrives i dette kapitlet, som "kontinuerlig", i betydning at det "glidende filteret" bare forskyves ett punkt fram i det digitaliserte signalet hver gang en ny beregning gjennomføres. Et alternativ ville være å forskyve waveleten med f.eks. halve waveletbredden.

Wavelettransformasjon brukes nesten utelukkende på diskrete signaler, siden beregningene er så omfattende at de nesten er umulige å gjennomføre analytisk (unntatt i svært enkle modell-beskrivelser).

For digitaliserte signaler (diskrete signaler) kan selve Morlet waveleten skrives som:

$$\Psi_{\omega_a,K,t_k}(t_n) = C\{\exp(-i\omega_a(t_n - t_k)) - \exp(-K^2)\} \cdot \exp(-\omega_a^2(t_n - t_k)^2/(2K)^2) \quad (14.8)$$

Her er det antatt at signalet vi skal analysere er beskrevet i ekvidistante punkter ved hjelp av tallrekken x_n for $n = 1...N$. Tiden t_k angir *midtpunktet til waveleten (!).*
Selve wavelettransformasjonen vil da kunne beskrives slik:

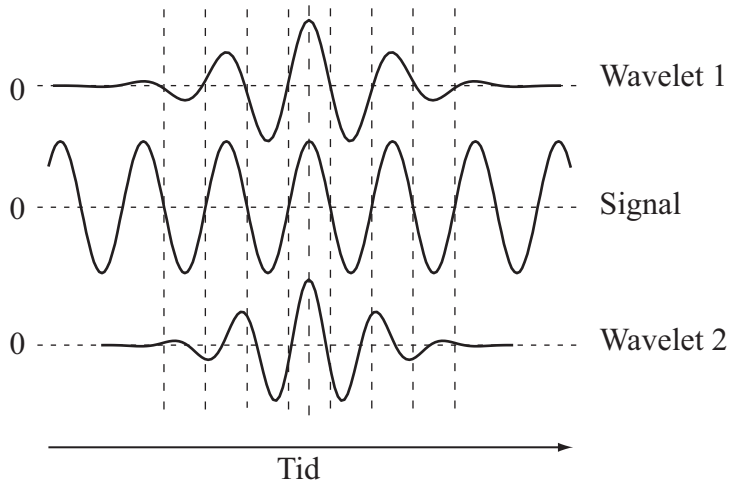
$$\gamma_K(\omega_a, t_k) = \sum_{n=1}^{N} x_n \Psi^*_{\omega_a,K,t_k}(t_n) \quad (14.9)$$

La oss bildeliggjøre prosessen litt for at vi skal bedre forstå hva den går ut på. I figur 14.3 viser vi et utsnitt av en tidsstreng sammen med to ulike valg av wavelets. Wavelettransformasjon består i å multiplisere signalet med waveleten, tidspunkt etter tidspunkt, og danne en sum av produktene.

For wavelet 1 ser vi at signalet skifter fortegn omtrent på samme sted som waveleten skifter fortegn. Det vil si at produktet i ethvert punkt blir positivt, og summen av produkter blir derfor ganske stor (svarer til at $\int \cos^2(\omega t)dt$ er positiv). For wavelet 2 skifter signalet fortegn på andre steder enn for waveleten. Noen av produktene blir derved positive og noen negative. Summen av produkter blir betydelig lavere enn for det første tilfellet (svarer til at $\int \cos(\omega_1 t)\cos(\omega_2 t)dt$ ofte er nær null når $\omega_1 \neq \omega_2$).

Vi har derved forsøkt å anskueliggjøre at wavelettransformasjonen av en regulær sinusbølge vil ha et maksimum når "bølgelengden" til waveleten svarer til "bølgelengden" til signalet i det tidsintervallet hvor vi foretar analysen.

For å analysere signalet x_n for andre bølgelengder, trenger vi å endre waveleten, og det gjøres blant annet ved hjelp av parameteren ω_a.



Figur 14.3: *Et sinusformet signal (i midten) sammen med en wavelet med samme periodetid (øverst) og en wavlet med noe kortere periodetid (nederst).*

14.4 Praktisk gjennomføring

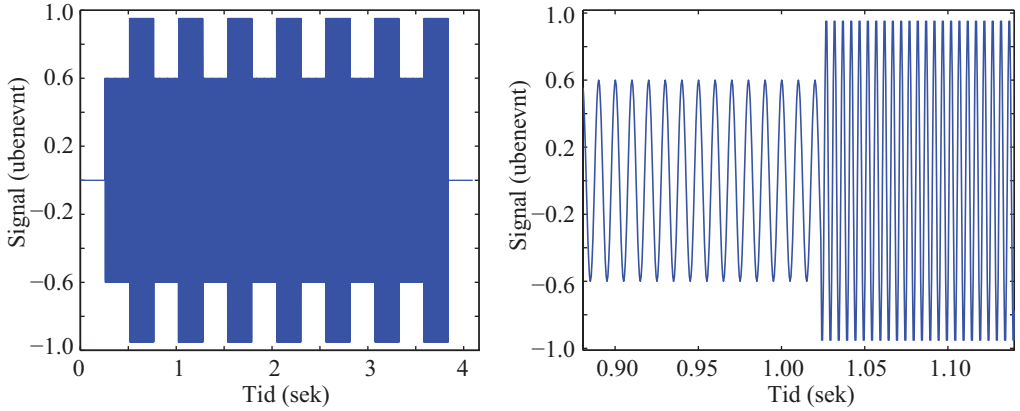
14.4.1 Eksempel på råmetoden for wavelettransformasjon

Vi skal nå vise i praksis et eksempel på råmetoden for wavelettransformasjon (gitt i ligning (14.9) og (14.8)). Vi tar også med Matlab-koden som er brukt for å generere de nødvendige figurene i tilfelle noen ønsker å forfølge eksemplet i litt andre retninger. Ellers er koden på de neste tre sidene nokså uinteressante.

Signalet vi genererer skifter mellom 100 og 200 Hz med faste intervaller (se figur 14.4). De ytterste bitene av signalet settes lik null. Merk at når vi genererer et signal med variabel frekvens i løpet av den tiden signalet eksisterer, *må* vi sørge for å unngå brudd i signalet akkurat i det tidspunktet frekvensen endrer seg. Vi får en god beskrivelse dersom vi tar utgangspunkt i *fasen* til signalet til enhver tid, og oppgraderer fasen for hvert nytt trinn i tidsbeskrivelsen. Dette er implementert i programkoden, og kontinuiteten i signalet er demonstrert i detaljutsnittet i figur 14.4.

```
function firkantSign5

% Genererer et signal som veksler med faste intervaller mellom to
% frekvenser, hver av disse har konstant amplitude. Nullstiller første og
% siste del av signalet. Oppintegrerer fasen ved genereringen for å unngå
% diskontinuiteter når signalet skifter fra en frekvens til den neste.
N = 4096*2;
f_sampl=2000.0;
T = N/f_sampl;
t = linspace(0,T,N);
delta_t = 1.0/f_sampl;
fase = 0.0;
omega1 = 200.0*2*pi;
omega2 = 100.0*2*pi;
k = 1;
for ii = 1:16
    if (mod(ii,2)>0) delta_fase = omega1*delta_t;
        A = 1.0;
    else delta_fase = omega2*delta_t;
        A = 0.6;
    end;
    for j = 1:512
        fase = fase+delta_fase;
        signal(k) = A*sin(fase);
        k = k+1;
    end;
end;
signal(1:512) = 0.0;
signal(N-512:N) = 0.0;
% % FOR UTTESTING (genererer et enkelt harmonisk signal):
% omeg = 2.0*pi*100;
% signal = sin(omeg*t);

% Plotter signalet, beregner så den fouriertransformerte og plotter denne
plot(t,signal,'-b');
xlabel('Tid (sek)');
ylabel('Signal (rel enheter)');
figure;
FTsignal = fft(signal);
f = linspace(0,f_sampl*(N-1)/N, N);
plot(f,abs(FTsignal),'-b');
xlabel('Frekvens (Hz)');
ylabel('Fourierkoeffisient (rel enhet)');
title('Den fouriertransformerte av signalet');
xlim([-0.0,300]);
% ylim([-0.2,2.2]);
```

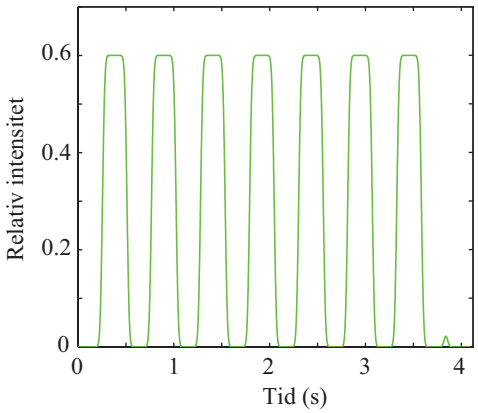
Vi implementerer så wavelettransformasjonen direkte ut fra ligningene (14.9) og (14.8). Analysefrekvensen for waveleten vi har valgt er $\omega_a = 2\pi 100$ (som svarer til 100 Hz i selve signalet). For hvert nytt punkt i wavelettransformasjonen som beregnes, forskyver vi toppunktet til waveleten fra å ligge helt i ytre høyre kant til helt i ytre venstre kant.



Figur 14.4: *Det genererte tidssignalet vi har brukt i eksemplet vårt. Venstre del viser hele signalet, mens en detalj av dette er vist i høyre del. Amplituden på 100 Hz signalet er 0.6 mens amplituden på 200 Hz signalet er 1.0.*

Resultatet er vist i figur 14.5. Vi ser at vi får en verdi på ca. 0.6 for de tidspunktene at det opprinnelige signalet hadde en frekvens lik analysefrekvensen.

En detalj er imidlertid verd å merke seg: Kurven i figur 14.5 har avrundete hjørner. Dette skyldes at waveleten har en endelig utstrekning i tid, og derfor vil "oppdage" en 100 Hz sekvens allerede før waveletens toppunkt er innenfor 100 Hz-området. Likeså vil waveleten "oppdage" områder hvor det ikke er 100 Hz selv når toppunktet til waveleten såvidt ligger innenfor 100 Hz-området.



Figur 14.5: *Den wavelettransformerte av tidssignalet for en analysefrekvens på 100 Hz. Parameteren K var 12, hvilket betyr at waveleten var grovt sett om lag 12 x (1/100) s = 0.12 s lang. Bredden på waveleten fører til avrunding av skarpe hjørner i diagrammet.*

Vi kommer tilbake til denne effekten i stor detalj siden.

Merk altså at en wavelettransformasjon slik vi har gjennomført den i dette eksemplet bare gir wavelet-transformasjonen for én analysefrekvens. Dersom vi ønsker å få med også andre

frekvenser, må dette gjøres ved å la prosedyren vi brukte her inngå i en løkke slik at alle ønskelige analysefrekvenser blir tatt med.

Det er imidlertid masse multiplikasjoner og summasjoner som skal gjøres i en slik rå-versjon av wavelettransformasjon. Det er smartere måter å gå fram på, og det skal vi se nærmere på nå, men vi gir resten av programkoden for råmetoden først:

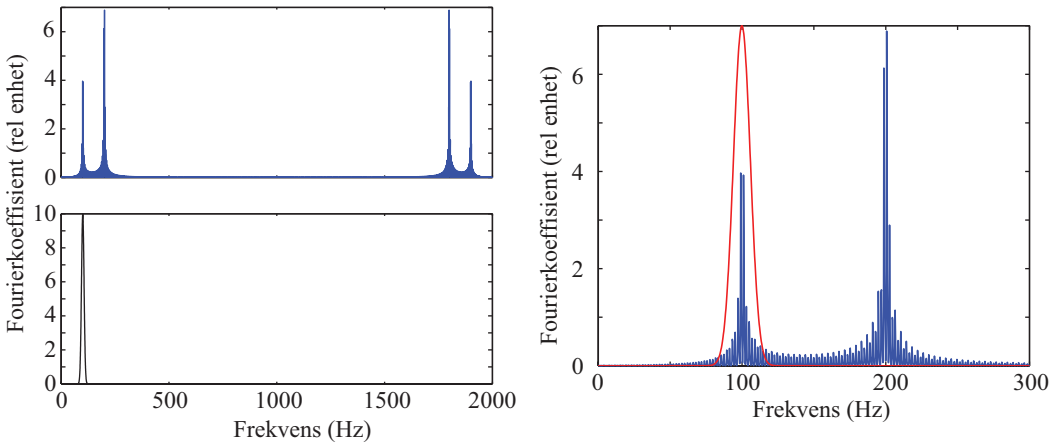
```
% Fortsettelse av forrige kodesnutt:
% Går nå over til å lage en Morlet wavelet for én valgt analysefrekvens
% for å gjennomføre en analyse ved rett-fram-metoden (tidkrevende)

f_analyse = 100.0;          % Frekvensen vi vil analysere for
omega_a = 2.0*pi*f_analyse; % Omega_analyse
K = 12;                     % Parameter som angir ca antall perioder
                            % innen omhyllingskurven
C = 0.7980*omega_a/(f_sampl*K); % Normeringskonstant (ulike valgmuligheter)
tx = linspace(-T,T,2*N+1);   % Lager én wavelet vi kan plukke ut deler av
harmSign = (cos(omega_a.*tx) + 1i.*sin(omega_a.*tx));   % Array for
                            % lagring av komplekst harmonisk signal
harmSign = C.*(harmSign - exp(-K*K)); % Trekker fra et lite korreksjonsledd
arg = tx.*omega_a/K;        % Array for gaussisk omhyllingskurve
arg2 = -0.5.*arg.*arg;
morlet = exp(arg2).*harmSign;   % Kombinerer for å lage Morlet wavelet

figure;                         % Plotter Morlet-waveleten om ønskelig
utsnitt = 240;                  % Plotter bare 2*utsnitt punkter av totalen
plot(tx(N-utsnitt:N+utsnitt),real(morlet(N-utsnitt:N+utsnitt)),'-b');
hold on;
plot(tx(N-utsnitt:N+utsnitt),imag(morlet(N-utsnitt:N+utsnitt)),':r');
xlim([-0.13,0.13]);    % Kan begrense plottet dersom man ønsker det, men
ylim([-0.048,0.048]);  % pass på at du ikke mister oversikten!
xlabel('Tid (s)');
ylabel('Relativ intensitet');
title('Morlet wavelet');

% Gjennomfører en waveletanalyse for den ene valgte analysefrekvensen (men
% waveleten morletT har sitt sentrum til "alle tider")
morletT = zeros(1,N);
for ii = 0:N-1
    morletT = morlet(N+1-ii:2*N-ii);
    produkt = morletT.*signal;
    integral = 0.0;
    for j = 1:N
        integral = integral + produkt(j);
    end;
    WT(ii+1) = integral;
end;
```

```
figure;         % Plotter wavelettransformasjonen for denne ene frekvensen
plot(t,abs(WT),'-g');
xlabel('Tid (s)');
ylabel('Relativ intensitet');
title('Wavelettransformen for én frekvens');
```

14.4.2 En mye mer effektiv algoritme

Da vi behandlet fouriertransformasjon i et tidligere kapittel, var vi såvidt inne på filtrering av signaler ved å maskere ut deler av et fourierspekter til et signal før vi transfomerer tilbake til en tidsbeskrivelse igjen.

I figur 14.6 viser vi øverst til venstre frekvensspekteret til signalet i figur 14.4. Nedenfor er det vist et frekvensspekter som har en gaussisk form og som er sentrert ved den ene av de to toppene i frekvensspekteret. Den gaussiske funksjonen viser ingen "folding" slik den fouriertransformerte til signalet gjør.



Figur 14.6: *Den fouriertransformerte av tidssignalet vårt, samme med en gaussformet funksjon som kan brukes ved en frekvensfiltrering. Se teksten for detaljer.*

Til høyre er det vist et utsnitt i frekvensskalaen, og den klokkeformede kurven er tegnet inn i samme diagram som den fouriertransformerte av signalet.

Poenget er da som følger:

Dersom vi multipliserer den fourieromvendte av signalet med den klokkeformede kurven, har vi bare tatt vare på frekvensbidrag i et relativt smalt frekvensområde sentrert rundt 100 Hz. Den delen som overlever inneholder informasjon om 100 Hz-delen av det opprinnelige signalet.

Dersom vi multipliserer den komplekse fourieromvendte av signalet (altså UTEN at absoluttverdien beregnes), vil vi beholde faseinformasjon i signalet. Det betyr at dersom vi

foretar en omvendt fouriertransformasjon, vil 100-Hz-signalet komme ut med en tidsangivelse som opprinnelig.

Det er på sett og vis denne type informasjon vi ønsker å få ut ved waveletanalyse! Men hvordan skal vi lage den gaussformede funksjonen? Her er vi heldige:

Den fouriertransformerte til en Morlet wavelet (ligning (14.8)) kan gis på følgende måte:

$$\hat{\Psi}_{\omega_a,K}(\omega) = 2\{\exp\left(-[K(\omega-\omega_a)/\omega_a]^2\right) - \exp(-K^2)\exp\left(-[K\omega/\omega_a]^2\right)\} \tag{14.10}$$

Vi ser at dette nettopp er en klokkeformet (gaussisk) funksjon (bortsett fra et ganske ubetydelig korreksjonsledd for de fleste valg av K). Toppunktet i klokkefunksjonen finnes ved analysefrekvensen.

Dette er et rent reelt uttrykk og er egentlig bare absoluttverdien av den fouriertransformerte av waveleten. Absoluttverdien av den fouriertransformerte av waveleten er den samme uansett hvilken tid toppunktet til Morlet waveleten er plassert.

Merk at den fouriertransformerte av en Morlet-wavelet bare har én topp i frekvensspekteret (ingen speiling!).

Det kan nå vises at wavelettransformasjon gjennomført med rå-metoden beskrevet ovenfor kan erstattes med følgende prosedyre:

- Beregn den fouriertransformerte av tidssignalet vi skal analysere.
- Beregn direkte den fouriertransformerte til en Morlet wavelet med den analysefrekvensen den vi er interessert i.
- Multipliser disse med hverandre, punkt for punkt.
- Foreta en invers fouriertransformasjon.
- Absoluttverdien av denne vil da gi informasjon om hvilke tidspunkt det opprinnelige signalet inneholdt frekvenser lik analysefrekvensen.

Den fouriertransformerte X_k av tidssignalet beregnes ut fra ligning (14.3). Denne transformasjonen behøver vi bare gjøre én gang.
Den fouriertransformerte til en Morlet wavelet med analysefrekvensen f_a kan skrives slik:

$$\hat{\Psi}_{f_a,K}(f_k) = 2\{\exp\left(-[K(f_k-f_a)/f_a]^2\right) - \exp(-K^2)\exp\left(-[Kf_k/f_a]^2\right)\} \tag{14.11}$$

Merk at det må være overensstemmelse mellom frekvensene f_k i den fouriertransformerte av signalet og den fouriertransformerte av waveleten. Ved å bruke den angitte prosedyren kan vi bygge opp en hel horisontal linje (alle tidspunkt) i én jafs. Ved å endre Morlet waveleten slik at den svarer til neste analysefrekvens, får vi bygget opp stadig nye horisontale linjer i waveletdiagrammet inntil vi har dekket så mange analysefrekvenser vi ønsker. Siden Fast Fourier Transform er så effektiv som den er, blir metoden vi nettopp har skissert mye raskere å gjenomføre enn rå-metoden vi omtalte ovenfor.

Programkode for å vise hvordan dette kan implementeres er gitt i det følgende. Koden bygger på de to forgående programsnuttene er kjørt forut for denne.

```
% Foretar til slutt en effektiv wavelettransformasjon basert på FFT/IFFT
% Beregner igjen FFT av tidsstrengen (holder å gjøre det én gang!)
FTsignal = fft(signal);
f = linspace(0,f_sampl*(N-1)/N, N);

% % Plotter frekvensspekteret dersom det ønskes
% figure;   % Plotter FFT (absoluttverdier only)
% nmax = floor(N/2);   % Plotter iblant bare litt av FFTen
% plot(f(1:nmax),abs(FTsignal(1:nmax)));
% xlabel('Frekvens (Hz)');
% ylabel('Relativ intensitet');
% title('Vanlig frekvensspektrum');

% Genererer (den foruriertransformerte av en wavelet) direkte, og plotter
% denne (de neste 11 linjene er bare tatt med for å kunne plotte wavelets)
fktr = (K/f_analyse)*(K/f_analyse);
FTwl = exp(-fktr*(f-f_analyse).*(f-f_analyse));
FTwl = FTwl - exp(-K*K)*exp(-fktr*(f.*f));   % Lite korreksjonsledd
FTwl = 2.0*FTwl;  % Faktor
figure;
plot(f,FTwl,'-k');
xlim([-0.0,300]);  % Kan begrense plottet dersom man ønsker det, men
% ylim([-0.2,2.2]);  % pass på at du ikke mister oversikten!
xlabel('Frekvens (Hz)');
ylabel('Relativ intensitet');
title('Den fouriertransformerte av waveleten');

% NÅ setter vi i gang med den virkelige effektive waveletanalysen!
M = 100;  % Antall frekvenser som skal inngå i analysen
fstart = 70.0;
fslutt = 300.0;
ftrinn = (fslutt/fstart)^(1/M);
f_analyse = fstart;
fbrukt = zeros(1,M);
% Loop over alle frekvenser som inngår i analysen
for jj = 1:M
   fktr = (K/f_analyse)*(K/f_analyse);
   FTwl = exp(-fktr*(f-f_analyse).*(f-f_analyse));
   FTwl = FTwl - exp(-K*K)*exp(-fktr*(f.*f));   % Lite korreksjonsledd
   FTwl = 2.0*FTwl;                             % Faktor
   % Beregner så en hel linje i scalogrammet i én jafs!
   WLdiagram(jj,:) = abs(ifft(FTwl.*FTsignal));
   %scalogram(jj,:) = sqrt(abs(ifft(FTwl.*FTsignal)));
   % Bruker denne siste varianten for å få svake partier bedre synlig
   fbrukt(jj) = f_analyse;        % Lagrer frekvensene som faktisk er brukt
   f_analyse = f_analyse*ftrinn;  % Beregner neste frekvens
end;
```

```
% Klargjør for for å kunne vise COI (cone of interest) i plottet
maxverdi = max(WLdiagram);
mxv = max(maxverdi);
% mxv  % Max verdi i WLdiagram. Brukes for å velge ok farge for markering
for jj = 1:M
    m = floor(K*f_sampl/(pi*fbrukt(jj)));
    WLdiagram(jj,m) = mxv/2;
    WLdiagram(jj,N-m) = mxv/2;
end;

% Plotter waveletdiagrammet
figure;
imagesc(t,log10(fbrukt),WLdiagram,'YData',[1 size(WLdiagram,1)]);
set(gca,'YDir','normal');
xlabel('Tid (sek)');
ylabel('Log10(frekvens i Hz)');
title('Wavelet Power Spektrum');
%title('Sqrt(Wavelet Power Spektrum)');  % Når sqrt av scalogram blir brukt
colorbar('location','southoutside');

input('Lukk alt');
close all;
```

14.5 Viktige detaljer

14.5.1 Faseinformasjon og skalering av utslaget

I vanlig fouriertransformasjon foretar vi i prinsippet to transformasjoner samtidig, nemlig en av typen

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\cos(\omega t)dt \tag{14.12}$$

og en av typen

$$X(\omega) = -i\int_{-\infty}^{\infty} x(t)\sin(\omega t)dt \tag{14.13}$$

Grunnen er at vi har både sinus og cosinusledd er at vi må kunne fange opp f.eks. et sinussignal i x uansett hvilken fase det har.

I vanlig fouriertransformasjon har vi *ett* startpunkt for anlysen. Det betyr at det er lett å finne hvilken relativ fase de ulike frekvenskomponentene har.

I kontinuerlig waveletanalyse har vi ulike startpunkt og lengder på analysevinduet underveis i beregningene. Det gjør det langt vanskeligere å holde orden på faser. Dette er nok en av grunnene til at vi nesten utelukkende bare tar utgangspunkt i absoluttverdien av wavelettransformasjonen i en eller annen variant når waveletanalysen skal angis. (Dersom

vi imidlertid skulle foreta en invers wavelettransformasjon etterpå, måtte vi selvfølgelig tatt vare på faseinformasjonen.)

Det er flere måter vi kan angi styrken i et waveletdiagram. Ofte brukes *kvadratet* av absoluttverdiene, hvilket gir *energien* i signalet.

Erfaringsmessig liker jeg ikke å bruke kvadratet av absoluttverdien, fordi forskjellen mellom de sterke og svake partiene da ofte blir så stor at vi mister informasjon om de svake partiene. Da er det ofte bedre å heller bruke absoluttverdien direkte ("amplitudenivå").

Jeg foretrekker imidlertid ofte å bruke *kvadratroten* av absoluttverdien. Da får jeg fram de svake partiene enda bedre enn om absoluttverdien ble plottet.

Vi står fritt i å velge hvordan resultatet fra wavelettransformasjonen blir plottet, men må ta følgen av vårt valg når vi skal hente kvantitative verdier ut av diagrammene.

14.5.2 Frekvensoppløsning vs tidsoppløsning

Vi skjønte ut fra figur 14.3 at når bølgelengden i signalet x er eksakt lik bølgelengden inne i waveleten, vil wavelettransformasjonen gi maksimal verdi. Endrer vi litt på bølgelengden i waveleten ved å endre på analysefrekvensen, vil transformasjonenen gi en lavere verdi, men likevel ikke null verdi. En waveletanalyse vil med andre ord ikke bare gi utslag for en frekvens som svarer til signalets, men også for nærliggende frekvenser.

Det er viktig å vite hvor langt ut denne "smitteeffekten" går, og det er tema for dette underkapitlet.

La oss da ta utgangspunkt i at en wavelettransformasjon innebærer en "digital filtrering" av et signal, slik vi anskueliggjorde i figur 14.6. Hvor skarp filteringen er bestemmes av bredden på den gaussiske klokkefunksjonen som brukes i filtreringen. Vi trenger da å finne en sammenheng mellom bredden i frekvensbildet og bredden av waveleten i tidsbildet.

I figur 14.7 viser til venstre tre ulike valg av waveleter (beregnet ut fra ligning (14.8)), og til høyre er den fouriertransformerte av waveleten (beregnet ut fra ligning (14.10)).

Vi vet fra tidligere at frekvensspekteret fra fouriertransformasjon av et sinussignal konvolutert med en gaussisk omhyllingskurve, selv har en gaussisk omhyllingskurve. Det får vi på ny bekreftet gjennom figur 14.7.

Bredden for tidsutstrekningen til waveleten kan bestemmes ved å ta utgangspunkt i omhyllingskurven (ut fra ligning (14.8)). Dersom vi definerer bredden som tidsforskjellen mellom toppunktet og et punkt hvor amplituden på omhyllingskurven har sunket til $1/e$ av maksimalverdien, er (halv)bredden:

$$\Delta t_{1/e} = 2K/\omega_a$$

Den tilsvarende bredden i den fouriertransformerte av waveleten er meget nær (ut fra ligning (14.10))

$$\Delta f_{1/e} = f_a/K = \omega_a/(2\pi K) \tag{14.14}$$

Det interessante er at

$$\Delta t_{1/e}\Delta f_{1/e} = (2K/\omega_a)\cdot(\omega_a/(2\pi K)) = 1/\pi$$

Dersom vi beregner "standardavviket" for tid og frekvens ut fra mer statistiske mål, slik:

$$\sigma_t^2 = \frac{\int t^2\Psi^2(t)dt}{\int \Psi^2(t)dt}$$

og

$$\sigma_f^2 = \frac{\int f^2\hat{\Psi}^2(f)df}{\int \hat{\Psi}^2(f)df}$$

kan det vises at

$$\sigma_t^2\sigma_f^2 = \frac{1}{2\pi} \tag{14.15}$$

Denne relasjonen er analog til Heisenbergs uskarphetsfunksjon. Eksempler i tråd med denne relasjonen er vist i figur 14.7.
Relasjonen er svært viktig for waveletanalyse. Dersom vi lar en wavelet strekke seg over en lang tid, vil bredden i frekvensdomenet være liten og visa versa. Med andre ord: *Vi kan ikke både få en nøyaktig tidsangivelse av detaljer i et signal samtidig som vi får en nøyaktig frekvensangivelse.*

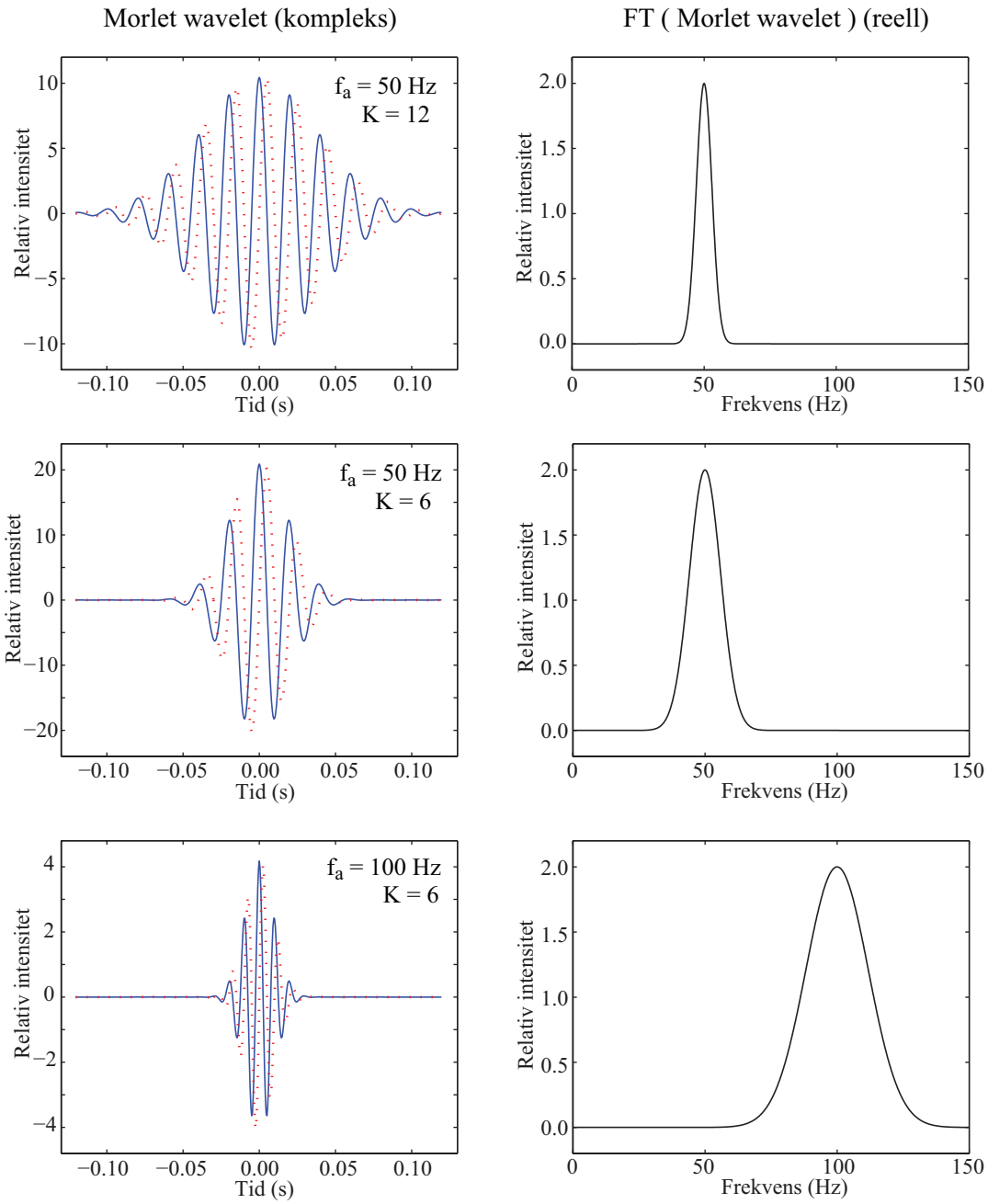
En interessant følge av ligning (14.14) er at

$$\Delta f_{1/e}/f_a = 1/K$$

Med andre ord, i en waveletanalyse holder vi oftest K konstant i hele analysen. Da er den relative usikkerheten i frekvensangivelsene konstant i hele diagrammet.
For å utnytte dette, er det normalt å velge en logaritmisk frekvensakse, i betydning at frekvensene vi velger å ta med i analysen forholder seg til hverandre som

$$(f_a)_{k+1} = (f_a)_k \cdot f_{faktor}$$

Vi har valgt logarismisk akse for de valgte analysefrekvensene i alle eksemplene i dette kapitlet, men det er selvfølgelig mulig å velge analysefrekvensene også ut fra en lineær skala, i alle fall dersom forskjellen mellom minste og største analysefrekvens er liten (f.eks. en faktor to eller mindre).



Figur 14.7: *Tre ulike wavelets som indikerer hvordan parmetrene analysefrekvensen ω_a og "bølgetallet" K virker inn på waveleten. En wavelet har en avgrenset utstrekning i tid (venstre del). Vi kan angi en bredde på omhyllingskurven f.eks. ut fra at verdien har sunket til $1/e$ av toppverdien.. Fourieromvendes denne waveleten, får vi frekvensresponsene vist til høyre. Legg merke til både posisjon i frekvesspekteret og bredden på de gaussformede kurvene. Bredden på frekvensresponsen kan angis på lignende måte som i tidsbildet. Det viktige er at breddene i tidsdomenet multiplisert med breden i frekvensdomenet er en konstant, hvilket innebærer at dersom den ene økes, vil den andre avta og visa versa.*

Sammenligning wavelets vs stykkevis FT

Skulle vi bruke stykkevis FT, ville et fast tidsintervall medføre at vi har svært få (eller ingen hele) periodetider innenfor intervallet for lave frekvenser, men ganske mange periodetider for høye frekvenser. Det betyr at vi ville få en elendig frekvensoppløsning for de laveste frekvensene (målt som relativ frekvens), men en langt bedre frekvensoppløsning for de høyere frekvensene. Det betyr at vi ville ende opp med en analyse som ikke ville være optimal.

Prosedyren som brukes i waveletanalyse gir en optimal tidsoppløsning for *alle* frekvenser. Men vi *kan* likevel velge å vektlegge tidsoppløsning *noe* på bekostning av frekvensoppløsning og omvendt alt etter hva vi ønsker å studere. Det gjør at metoden blir et meget slagkraftig hjelpemiddel i mange sammenhenger.

Kunne vi valgt en stykkevis FT der vi faktisk brukte skaleringsprinsippet med å stykke opp tidsstrengen i mindre biter når vi analyserte høye frekvenser enn ved lave? Det ville kreve en enorm mengde fouriertransformasjoner, men ville da gi omtrent samme resultat som en waveletanalyse. Resultatet ville likevel ikke bli like godt. Vi ville nemlig med skalerte FT-intervaller i prinsippet hatt en ren waveletanalyse, men med en wavlet som har en annen form enn Morlet. Det ville være en firkantpuls som da var omhyllingskurven. Frekvensresponsen ville da ha en form

$$\left(\frac{\sin(x)}{x}\right)^2$$

i stedet for en gaussisk omhyllingskurve. Resultatet ville bli en hale utenfor den sentrale toppen som gjør at vi får en dårligere frekvensbestemmelse enn ved Morlet wavelets.

I denne sammenheng kan vi trekke analogier til diffraksjon. Dersom vi sender lys inn mot en smal spalt og har *samme* intensiteten over hele spalten, vil diffraksjonsintensiteten nettopp være gitt som $(\frac{\sin(\theta)}{\theta})^2$. Dersom vi derimot sendte en lysstråle med *gaussisk* intensitetsprofil gjennom spalten, ville vi fått en gaussisk intensitetsprofil på skjermen uten ekstra striper ved siden av den sentrale toppen.

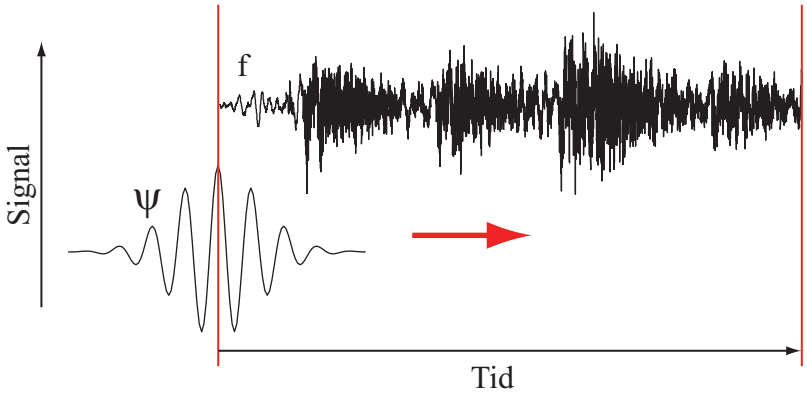
14.5.3 Randproblem

Når vi foretar en wavelettransformasjon, multipliserer vi i prinsippet et signal med en wavelet og summerer alle produktene. Vi flytter så waveleten og gjør det samme på ny. Dette gjentas om igjen og om igjen fra den situasjonen at waveletens midtpunkt ligger helt i den ene enden av signalet til waveletens midtpunkt ligger i andre enden av signalet.

Vi endrer så analysefrekvensen for waveleten og gjør det samme på ny.

Her oppstår det imidlertid et problem. Så lenge wavleten ikke er fullstendig innenfor dataområdet, vil vi måtte forvente et annet resultat enn om hele waveleten ble brukt i beregningene. Dette er anskueliggjort i figur 14.8. For den posisjonen waveleten har i forhold til dataene i denne figuren, vil bare om lag halvparten av waveleten benyttes i praksis. Det betyr at summen av produktene må forventes å bli langt lavere (i størrelsesorden halvparten) av hva summen ville blitt dersom vi hadde fullt overlapp.

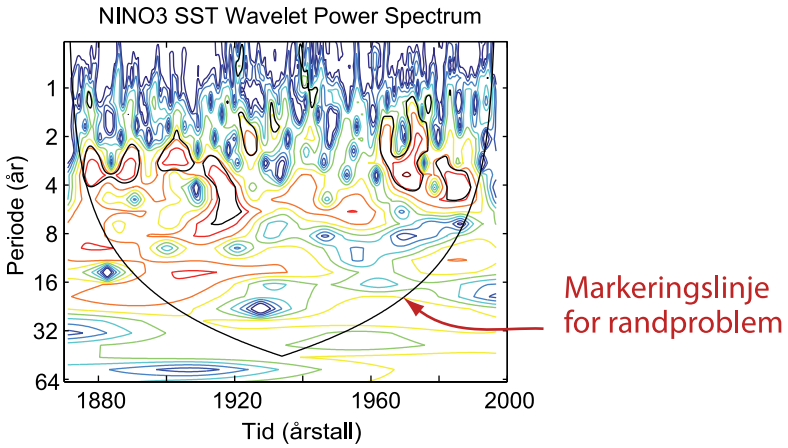
Av den grunn kan vi ikke stole noe særlig på datapunktene i en endelig wavelettransformasjon dersom waveleten som er brukt ikke ligger fullstendig innenfor datastrengen vi



Figur 14.8: *Det er ikke mulig å få et korrekt waveletresultat for tider og frekvenser der ikke hele waveleten kommer innenfor dataområdet under beregningene.*

analyserer. Det er derfor vanlig å markere ytterområdet mhp tid hvor vi har et randproblem.

I figur 14.9 er det vist et eksempel på et waveletdiagram etter analyse av temperaturoscillasjoner i det sydlige Stillehav. Det er brukt kvoter sammen med farger for å markere "energien" i ulike former for svingninger (periodisitet) etter som de har utviklet seg de siste vel hundre år.



Figur 14.9: *Eksempel på et waveletdiagram for temperaturoscillasjoner i det sydlige stillehav. Figuren er produsert med data og programvare tilgjengelig fra http://paos.colorado.edu/research/wavelets/ tilgjengelig april 2013.*

I dette diagrammet er det tegnet inn en krum V-formet linje som starter langt nede og midt i waveletdiagremmet med symmetriske buede linjer som går opp og ut mot kanten. Disse linjene markerer området hvor det aller meste av waveletene er fullstendig innenfor datastrengen: Alt over denne buede V-en er gyldige data. Alt under streken er data vi må ta med en klype salt.

I programeksemplene gitt i dette kapitlet har vi valgt å legge inn en markering som svarer til at bare den ytre delen av waveleten som har verdi mindre enn $1/e$ av maksimum på omhyllingskurven ligger utenfor diagrammet. Vi har bare med et så lite frekvensområde at vi i egne eksempler ikke får fram hele den buete V-en, men bare et smalt horisontalt bånd av den totale V-en. Alle deler av waveletdiagrammet som ligger mellom disse markeringene har kun ubetydelige feil på grunn av randproblematikken. Vi gir detaljer nedenfor om hvordan markeringene settes opp.

14.6 Optimalisering

Waveletanalyse er mer krevende enn vanlig fouriertransformasjon. Vi må velge hva slags wavelet vi vil bruke. Selv om vi holder oss til Morlet wavelets, må vi bestemme oss for hvilket "bølgetall" vi vil bruke.

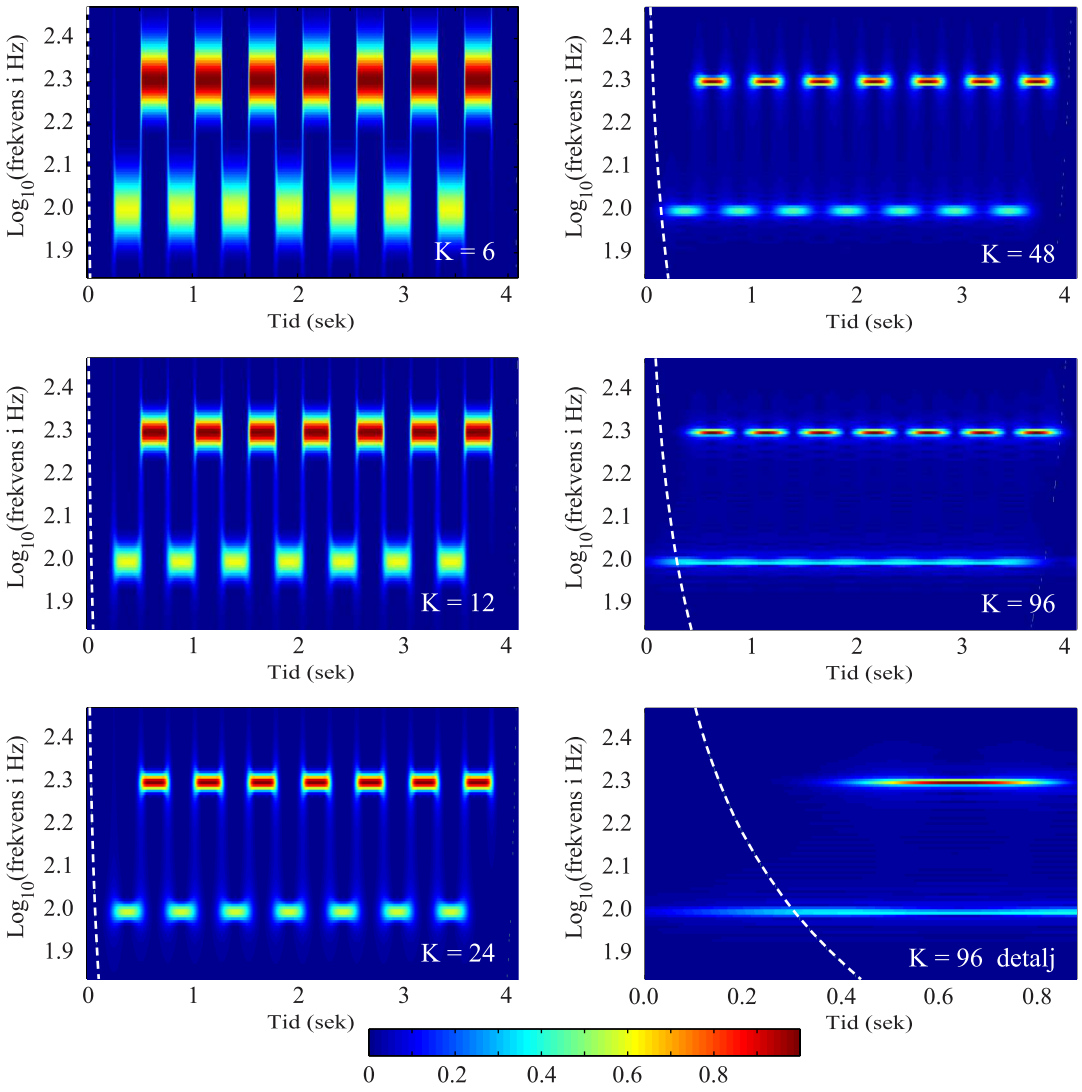
Vi har tidligere sett at ved å øke bølgetallet K, vil waveleten ha en betydelig verdi over et større tidsrom enn ved lavt bølgetall (ved samme analysefrekvens). Videre har vi sett at når "bredden" på waveleten i tidsdomenet er stor (det vil si stor K-verdi), vil "bredden" på den fouriertransformerte av waveleten bli liten. Produktet av bredden på waveleten i tidsdomenet og bredden av waveleten i frekvensdomenet er jo konstant.

Følgen er at det finnes ingen Ole Brumm-løsninger: "Ja takk, begge deler" innen waveletanalyse. Ønsker vi å få en nøyaktig angivelse av tidsforløp, vil små K-verdier være å foretrekke. Ønsker vi å få så nøyaktige frekvensangivelser som mulig, bør K-veriden være høy. Vi ønsker i prinsippet så god tidsoppløsning og frekvensoppløsning som mulig, men må alltid velge et kompromiss.

Det optimale resultatet oppnås ofte dersom vi tar utgangspunkt i signalet selv. Signalet har ofte innebygget en uskarphet i tid og/eller en uskarphet i frekvens. Vi kan aldri få en bedre oppløsning i tid ved waveletanalyse enn den oppløsningen signalet selv har. Tilsvarende for analyse av frekvens.

I figur 14.10 er det vist waveletdiagrammer av signalet vi drøftet ovenfor som vekslet mellom 100 og 200 Hz. Fem ulike bølgetall K er brukt. Vi ser at for lave K-verdier er tidsoppløsningen meget presis, men frekvensbestemmelsen er elendig. For høye K-verdier er det motsatt: Frekvensoppløsningen er god, men tidsoppløsningen er elendig.

I dette tilfellet er det egentlig ikke mye mer å hente i frekvensoppløsning når vi går fra $K = 48$ til $K = 96$. Det betyr at frekvensoppløsningen i signalet selv (siden varigheten av hver periode med "konstant frekvens") svarer omtrent til oppløsningen vi oppnår med en K-verdi litt større enn 48. (Strengt tatt er det forskjellig frekvensoppløsning på 100 Hz- signalene og 200 Hz signalene siden det er ulikt antall perioder innen hver av disse periodene med konstant frekvens.)



Figur 14.10: *Waveletdiagrammer for tidssignalet i figur 14.4 for seks ulike "bølgetall" K. Se teksten for detaljer.*

I figur 14.10 har vi gjort markeringen av venstre side av randproblemsområdet ekstra tydelig. Vi ser da klart at randproblemet øker med økende K-verdi. Det kan være interessant å merke seg at randproblemmarkeringen endrer seg med analysefrekvensen. Videre er det nyttig å vite at avstanden fra sidekanten til randproblemmarkeringen også indikerer utsmøring i klare tidsangivelser i waveletdiagrammet. All tidsinformasjon i analysen blir smurt ut om lag med en tidsforskjell som nettopp svarer til avstanden fra randen til randproblemmarkeringen.

Hva er da "beste valg" av alle analysene gitt i figur 14.10? Vel, det kommer an på hva vi ønsker å få fram av opplysninger. Diagrammet for $K = 6$ demonstrerer at endringen fra 100 til 200 Hz (og omvendt) foregår meget skarpt i tid. Diagrammet for $K = 96$ viser at frekvensen er så ensartet som den kan være innenfor hver av tidsintervallene. Skulle vi velge en slags generell optimalisering, kunne kanskje $K = 48$ eller deromkring være et godt valg.

Optimalisering i frekvensoppløsning (programmeringsteknisk)

En annen form for optimalisering ligger i valg av frekvensområde for analysen. I en digital fast fourier analyse får vi automatisk "alle" frekvenser mellom null og samplingsfrekvensen (men bare halvparten er nyttig pga folding). For en kontinuerlig waveletanalyse velger vi oftest å innskrenke frekvensområdet til det området der frekvensinnholdet er av interesse.

I figur 14.10 valgte vi bare å ta med frekvenser mellom 70 og 300 Hz i analysen. Grunnen er at vi visste at signalet bare inneholdt frekvenser nær opp til 100 og 200 Hz. Det kan ofte være en fordel å starte med en vanlig fouriertransformasjon for å sikre oss at vi velger et frekvensområde som egner seg.

Det er imidlertid viktig også å tenke på hvor mange mellomliggende frekvenser vi skal ta med i analysen. I denne sammenheng må vi gå tilbake til "bredden" av wavleten i frekvensdomenet. Denne bredden var som vi tidligere har sett:

$$\Delta f = f_a/K$$

Denne "bredden" var bestemt ved at den gaussformede frekvenskurven var kommet ned til $1/e$ av maksimum verdi. Vi ønsker ikke å gå så store steg i frekvens fra en analysefrekvens til den neste, men kanskje bare en brøkdel av dette.

Praktisk testing viser at et optimalt valg av forskjellen mellom en analysefrekvens og den neste er da om lag

$$f_{a,neste} - f_{a,naa} = f_{a,naa}/8K \tag{14.16}$$

Dersom vi skal spenne over et frekvensintervall $[f_{start},\ f_{slutt}]$ kan det da enkelt vises at vi bør bruke M analysefrekvenser i en logaritmisk orden, hvor

$$f_{slutt} = (1 + \frac{1}{8K})^{M-1} \cdot f_{start}$$

Antall analysefrekvenser er da:

$$M = 1 + \log(f_{slutt}/f_{start})/\log(1 + \frac{1}{8K}) \tag{14.17}$$

Optimalisering i tidsoppløsning (programmeringsteknisk)

Et kontinuerlig waveletdiagram kan iblant bestå av svært mange punkter. Dersom vi f.eks. tar utgangspunkt i lyd som er digitalisert med en samplingsfrekvens på 44.1 kHz, og vi studerer lyd med frekvens i området 100 - 10 000 Hz, kan vi i praksis plukke ut bare hvert fjerde punkt i tidsdimensjonen uten problemer. Når vi atpåtil vet at wavleten har en bredde på om lag K ganger periodetiden for analysefrekvensen, innser vi at vi kan fjerne enda flere punkter i tidsdimensjonen uten at det vil oppdages i et waveletdiagram.

Det kan iblant være av interesse å optimalisere et waveletdiagram mhp tidsangivelsen. Ikke minst er dette viktig for å få plottefiler som er små små at de lett lar seg innlemme i rapporter og liknende.

I praksis viser det seg at vi kan nøye oss med å gjengi hvert P-te punkt i tidsdimensjonen i et waveletdiagram uten at nevneverdig informasjon går tapt når P er gitt ved:

$$P = HeltallsverdienAv\left(\frac{K}{24}\frac{f_s}{f_{a,max}}\right) \tag{14.18}$$

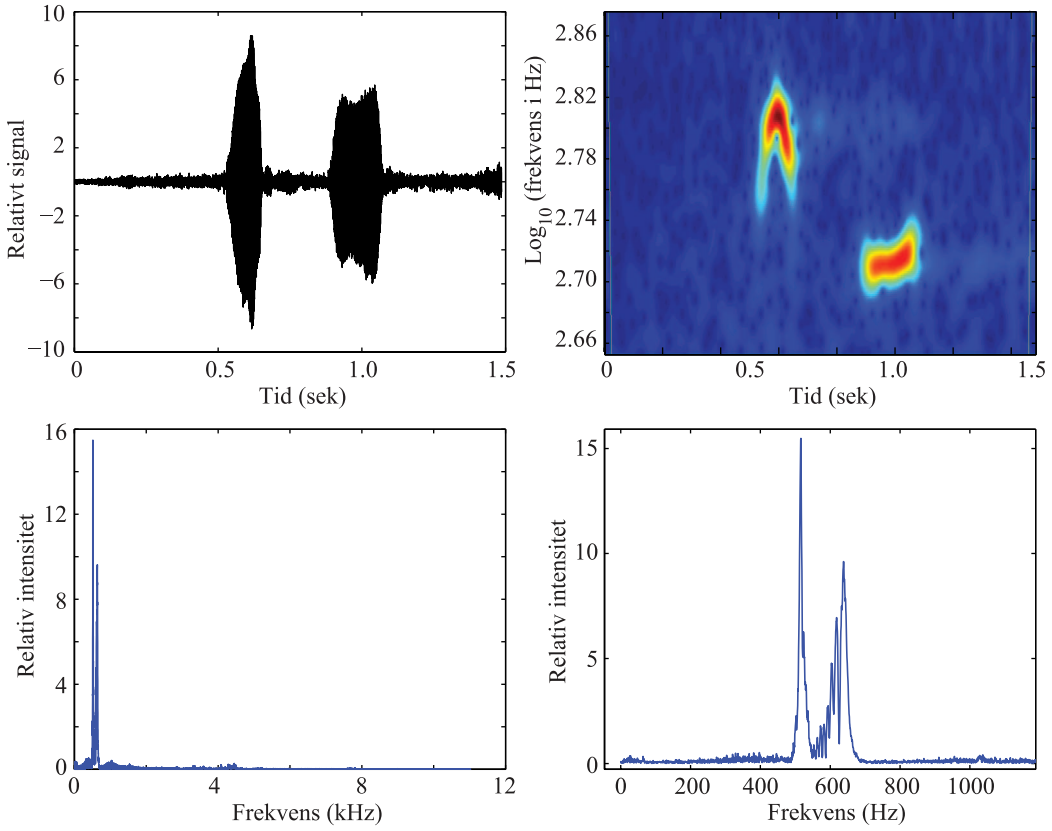
14.7 Et realistisk eksempel

Figur 14.11 viser et eksempel på en optimalisert waveletanalyse. Signalet er en lydfil i CD-kvalitet som gir lyden av en gjøk som synger sitt "ko ko". Signalet er gitt i tre varianter, nemlig som et rent tidssignal, som frekvensspekter etter en vanlig FT, og endelig analysert ved å bruke wavelettransformasjon.

I den totale arbeidsgangen er første trinn å velge ut et passe utsnitt fra lydfilen. Det gjøres

ved at vi velger startpunkt og totalt antall punkter som skal hentes ut fra den tilgjengelige datafilen. Dernest foretas en fourieranalyse. Fra fourierspekteret ser vi at lyden stort sett bare inneholder frekvenser mellom 450 og 750 Hz. Av den grunn er waveletanalysen begrenset til dette frekvensintervallet.

Til slutt må vi forsøke med ulike K-verdier og velge et "beste" kompromiss mellom god tidsangivelse og frekvensangivelse samtidig. Vi må da bestemme oss for hvorvidt vi vil prioritere tidsoppløsning (ved å ha en liten K), men da på bekostning av en nokså bred frekvensrespons, eller godta en litt dårligere tidsoppløsning (ved å velge en større K) for å få en noe bedre frekvensoppløsning. Hva som er optimalt kommer an på selve signalet vi analyserer og vil avhenge av hva den enkelte vektlegger mest. I vårt eksempel ble $K = 40.0$ benyttet.



Figur 14.11: *Gjøkens "ko ko" analysert i tidsbildet, i frekvensbildet (med et utsnitt) og i kombinasjonen: Frekvens og tid i form av et waveletdiagram.*

Legg merke til de nydelige detaljene som kommer fram i waveletanalysen. Har du f.eks. vært klar over at lyden i det første "ko"-et faktisk endrer seg betydelig den korte stunden lyden varer? Detaljer i waveletanalyse av fuglesang gjør det mulig for ornitologer å gjenkjenne fugler individuelt. Detaljene er finere enn hva en menneskehørsel klarer å oppfatte.

Det bør være innlysende at wavelettransformasjon gir langt mer interessante data enn en

vanlig fouriertransformasjon for en slik type lyd.

Vi har tatt med et Matlab program som kan brukes for å analysere biter av lydfiler. Programmet kan kopieres direkte fra pdf-filen slik at du slipper å skrive den inn på nytt.

[♠ ⇒ Noen kommentarer:
Programmet er selvsnekret og bærer preg av at når jeg selv gjør programmering, er det for å få gjort en jobb. De av dere som er mer finesmackere i programmering får heller lage deres eget program som er mer i tråd med moderne trender og krav. Selv startet jeg med hullkort og hadde spesialtillatelse til å bruke hele 150 kB av RAM av totalt 250 kB (!) i Norges største datamaskin på den tiden (Regnemaskinen Blindern/Kjeller) til x antall millioner kroner. Jeg er derfor fornøyd når et program fungerer, selv om koden ikke er så elegant som den kunne vært! ⇐ ♠]

```
function WaveletTransform7

% Program som foretar en waveletanalyse av et lydsignal som foreligger som
% en wav-fil. Laget for FYS2130 våren 2013 av Arnt Inge Vistnes.
% Denne versjonen er fra 18042013.

% Navn på noen lydfiler (her angitt sammen med egnede parametre for analyse)
c = 'Svarttrost2.wav';  % Nstart 17000, 64 k, 1500-8000 Hz, K12-96
% c = 'bokfink.wav';    % Nstart 34 000, 128 k, K 48, 2000-8000 Hz (minst)
% c = 'gjok.wav';       % Nstart 12000, 450-750 Hz + noe støy ved > 4000 Hz

% PARAMETRE VI MÅ SETTE:  (Parametre for waveletanalysen settes nedenfor!)
N = 1024*64;            % Lengde på data-utsnitt (lydfil) (helst 2^n) Her: "64k"
nstart = 17000;         % Startpunkt for utsnitt fra lydfil

% Leser passe utsnitt av lydfil, spiller den av og plotter tidsbildet
nslutt = nstart+N-1;
[y, fs] = audioread(c, [nstart nslutt]); % Les array y(N,2) fra fil
% 'fs' er vanligvis 44100 (samplingsfrekvens ved CD kvalitet)
h = zeros(N,1);     % Plukker ut bare én kanal fra stereosignalet lest
h = y(:,1);
sound(h,fs);        % Spiller av utsnittet som er brukt
T = N/fs;           % Total tid lydutsnittet tar (i sek)
t = linspace(0,T*(N-1)/N,N);
plot(t,h,'-k');
title('Opprinnelig tids-funksjon');
xlabel('Tid (sek)');
ylabel('Signal (rel enhet)');

%**************************************************************************

% Foretar så en effektiv wavelettransformasjon basert på FFT/IFFT. På dette
% punktet må totalt antall punkter N og samplingsfrekvens fs være kjent.
```

```
% Beregner først FFT av tidsstrengen h (gjøres bare én gang!)
FTsignal = fft(h);

% Plotter frekvensspekteret (absoluttverdier only)
f = linspace(0,fs*(N-1)/N, N);
nmax = floor(N/2);          % Plotter bare opp til halve samplingsfrekv.
figure;
plot(f(1:nmax),abs(FTsignal(1:nmax)));
xlabel('Frekvens (Hz)');
ylabel('Relativ intensitet');
title('Vanlig frekvensspektrum');

% Dernest selve waveletanalysen

% INPUT PARAMETRE vi selv må velge følger her:
K = 12;             % Morlet-wavelet-bredde (kan være 6 - 400)
fmin = 1500.0;      % Minimum frekvens i waveletanalysen (i Hz)
fmax = 8000.0;      % Maximum frekvens

% Beregner # analysefrekvenser, skriver til skjerm, klargjør frekvensene
M = floor(log(fmax/fmin) / log(1+(1/(8*K)))) + 1;
AntallFrekvenserIAnalyse = M
ftrinn = (fmax/fmin)^(1/(M-1));
f_analyse = fmin;

% Allokerer plass til waveletdiagrammet og array for lagring av frekvenser
WLdiagram = zeros(M,N);
fbrukt = zeros(1,M);

% Løkke over alle frekvenser som inngår i analysen
for jj = 1:M
   faktor = (K/f_analyse)*(K/f_analyse);
   FTwl = exp(-faktor*(f-f_analyse).*(f-f_analyse));
   FTwl = FTwl - exp(-K*K)*exp(-faktor*(f.*f));   % Lite korreksjonsledd
   FTwl = 2.0*FTwl;                               % Faktor (ulike valg!)
   % Beregner så en hel linje i waveletdiagrammet i én jafs!
   %WLdiagram(jj,:) = abs(ifft(FTwl.*transpose(FTsignal)));% Ett alternativ
   WLdiagram(jj,:) = sqrt(abs(ifft(FTwl.*transpose(FTsignal)))); % Ett annet
   % Bruker den siste varianten for å få svake partier bedre synlig
   fbrukt(jj) = f_analyse;        % Lagrer frekvensene som faktisk er brukt
   f_analyse = f_analyse*ftrinn;  % Beregner neste frekvens
end;

% Reduserer filstørrelse ved å fjerne mye av overflødig informasjon i tid.
% Dette gjøres kun for at filstørrelsen på plottene skal bli håndterbar.
P = floor((K*fs)/(24 * fmax));  % Tallet 24 kan endres ved behov
TarBareMedHvertXITid = P
NP = floor(N/P);
AntallPktITid = NP
```

```
for jj = 1:M
    for ii = 1:NP
        WLdiagram2(jj,ii) = WLdiagram(jj,ii*P);
        tP(ii) = t(ii*P);
    end;
end;

% Foreta en markering i plottet for å vise områder med randproblemer
maxverdi = max(WLdiagram2);
mxv = max(maxverdi);
for jj = 1:M
     m = floor(K*fs/(P*pi*fbrukt(jj)));
     WLdiagram2(jj,m) = mxv/2;
     WLdiagram2(jj,NP-m) = mxv/2;
end;

% Plotter waveletdiagrammet
figure;
imagesc(tP,log10(fbrukt),WLdiagram2,'YData',[1 size(WLdiagram2,1)]);
set(gca,'YDir','normal');
xlabel('Tid (sek)');
ylabel('Log10(frekvens i Hz)');
%title('Wavelet Power Spektrum');       % Velg denne når det er aktuelt,
title('Sqrt(Wavelet Power Spektrum)');  % men denne når sqrt blir brukt
colorbar('location','southoutside');

input('Lukk alt');
close all;
```

14.8 To ytterligere eksempler

Vi tar med to ytterligere eksempler på waveletanalyse. Det første er liknende den vi hadde for gjøkens ko ko. Vi har valgt kvitringen til en bokfink, som dominerer fuglesangen i april. Bokfinkens sang blir karakterisert på flere ulike måter. Selv liker jeg best karakteristikken "*tit tit tit tit tit.... el-ske-de-viv*". Det morsomme med waveletanalysen er at lydbildet er langt mer komplisert enn det vi mennesker oppfatter. Det er en meget rask variasjon i frekvens innen hver "tit" som vi ikke oppfatter. K-verdi brukt i analysen var 48.0.

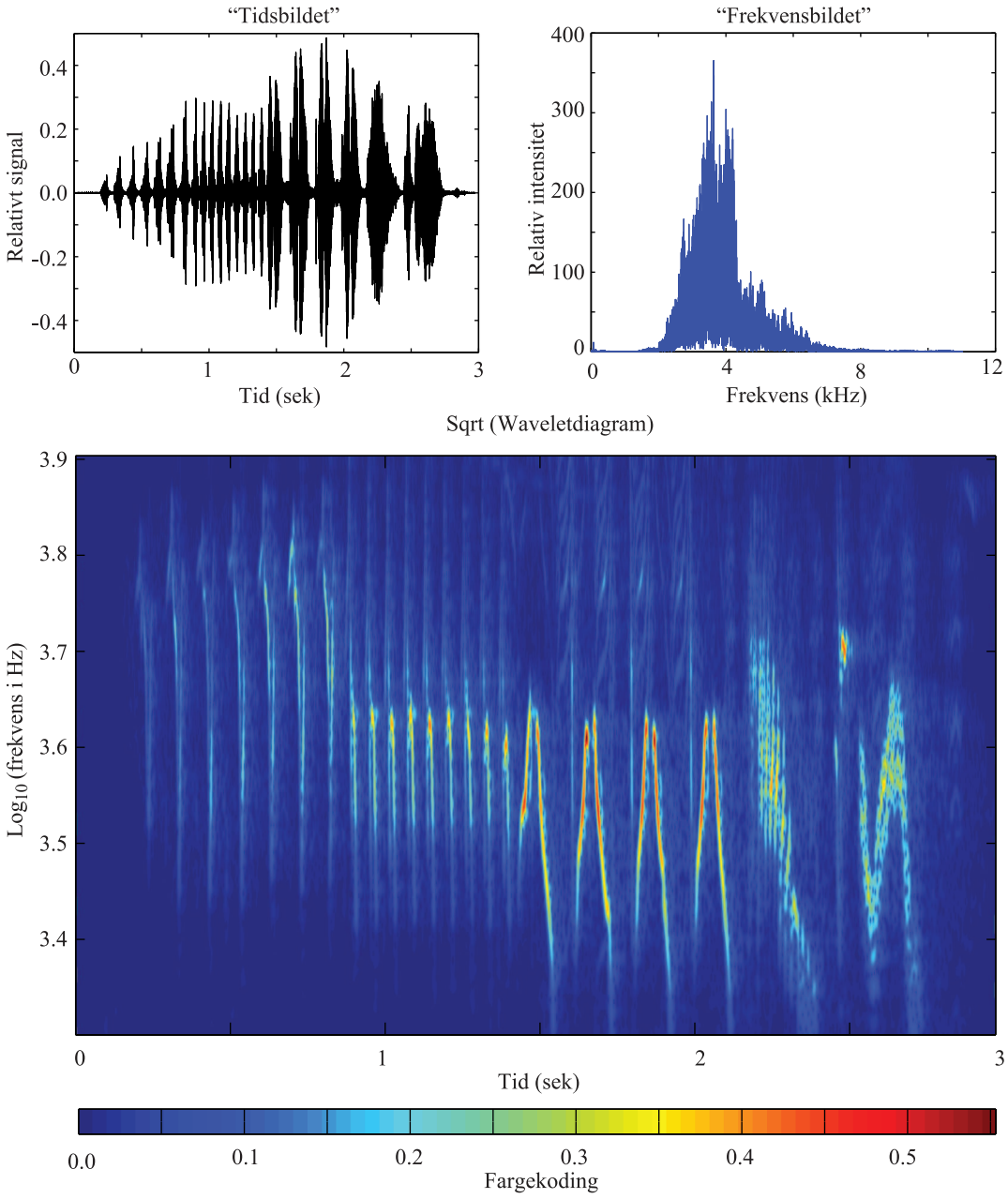
Det andre eksemplet er en waveletanalyse av en trompetlyd. Vi har valgt et utsnitt i tid der trompeten holder samme tone, og intensiteten på lyden opplever vi som temmelig konstant. Tidsbildet av lyden viser mer variasjon i intensitet enn det vi oppfatter. Frekvensbildet (frekvensspekteret) er imidlertid helt slik vi hadde forventet det. Det består av en rekke skarpe linjer som viser grunntonen og de harmoniske. Frekvensaksen er lineær, og derved er avstanden mellom to nærliggende harmoniske fast, nærmere bestemt lik frekvensen til grunntonen.

En waveletanalyse av et slikt signal klarer ikke å matche skarpheten i frekvensspekteret, så dersom vi først og fremst er interessert i frekvensen til grunntonen og de harmoniske *for en vedvarende tone*, er fourieranalyse metoden som bør velges.

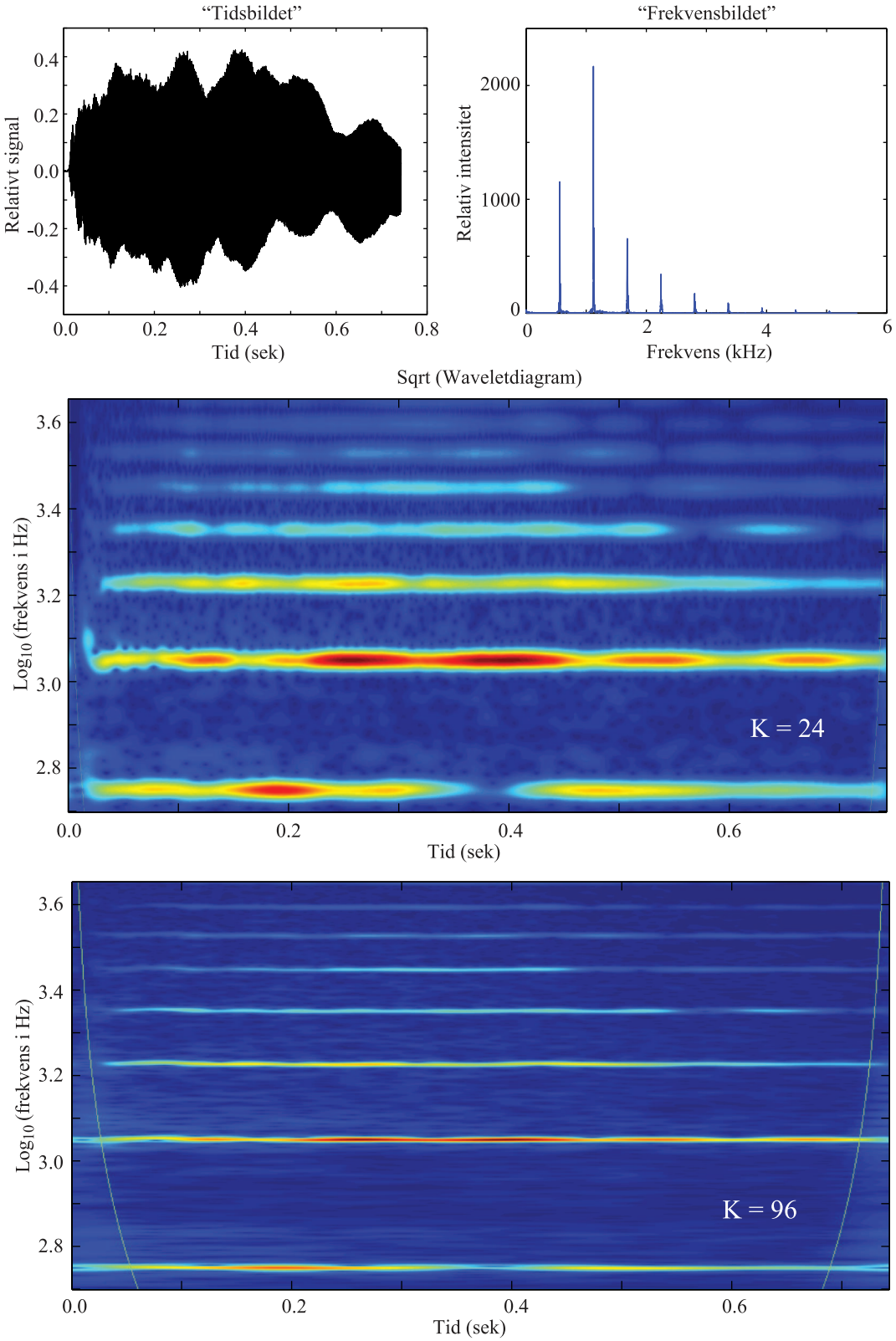
Dersom vi er interessert i variasjoner i lyden over tid, egner imidlertid ikke fourieranalysen seg. Da kommer waveletanalysen inn. Vi har tatt med to ulike varianter av analyse, basert på bølgetallene $K = 24$ og $K = 96$. I det første tilfellet er frekvensoppløsningen nokså dårlig, men tidsoppløsningen passe i forhold til signalet. I det siste tilfellet er frekvensoppløsningen god, men tidsoppløsningen dårlig.

Får vi noe ekstra ut av waveletanalysen framfor fourieranalysen? Ja, faktisk. Vi ser at styrken på grunntonen og de harmoniske varierer litt i tid. Vi ser også at det er en viss veksling mellom intensiteten til grunntonen og den første harmoniske: Når den ene er kraftig er den andre svak og visa versa. Dette gir liv til lydbildet, og viser et eksempel på at det er vanskelig å erstatte virkelig lyd med syntetisk lyd.

Legg forøvrig merke til at avstanden mellom de harmoniske ikke er konstant i et normalt waveletdiagram, siden vi der normalt har en logaritmisk frekvensakse.



Figur 14.12: *Bokfinkens "tit tit tit tit tit.... el-ske-de-viv" analysert i tidsbildet, i frekvensbildet og i en wavelettransformasjon.*



Figur 14.13: *En ren trompetlyd analysert i tidsbildet, i frekvensbildet og i en wavelettransformasjon. To ulike K-verdier er brukt i waveletanalysen.*

14.9 Wavelet-ressurser på nett

1. A-H Najmi og J Sadowsky: "The continous wavelet transform and variable resolution time-frequency analyses." Johns Hopkins APL technical digest, vol 18 (1997) 134-140. Tilgjengelig på http://www.jhuapl.edu/techdigest/TD/td1801/najmi.pdf den 15. april 2013.

2. http://www.polyvalens.com/blog/ , "A really friendly guide to wavelets", med mere (C. Valens). Tilgjengelig 15. april 2013.

3. http://tftb.nongnu.org/ , "Time-frequency toolbox". Tilgjengelig 15. april 2013.

4. http://dsp.rice.edu/software/rice-wavelet-toolbox , "Rice Wavelet Toolbox." Tilgjengelig 15. april 2013.

5. http://www.cosy.sbg.ac.at/˜ uhl/wav.html , Mengde wavelet-lenker. Tilgjengelig 15. april 2013.

6. Et 72 siders hefte av Liu Chaun-Lin: "A tutorial of the wavelet transform" (datert 23. februar 2010) er tilgjengelig på http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf tilgjengelig 15. april 2013. Heftet tar også for seg wavelets brukt i bildebehandling.

14.10 Læringsmål

Etter å ha jobbet deg gjennom dette kapitlet bør du kunne:

- Gjøre rede for likheter og forskjeller mellom fouriertransformasjon og wavelettransformasjon.
- Gjøre rede for hvilke signaler fouriertransformasjon er å foretrekke og hvilke signaler der wavelettransformasjon foretrekkes. Begrunn hvorfor.
- Forklare hva vi kan lese ut av et gitt waveletdiagram.
- Forklare hvordan vi kan justere en wavelettransformasjon for å fremheve detaljer i tid, eller detaljer i frekvens.
- Forklare kvalitativt analogier mellom wavelettransformasjon og Heisenbergs uskarphetsrelasjon.
- Bruke et waveletanalyseprogram og optimalisere analysen.

14.11 Oppgaver

Forståelses- / diskusjonsspørsmål

1. Hva er viktigste forskjell mellom fouriertransformasjon og waveletanalyse?
2. I hvilke situasjoner gir fouriertransformasjon et temmelig ubrukelig resultat?
3. Hvilke ulemper har wavelettransformasjon sammenlignet med fouriertransformasjon?
4. Når ble fouriertransformasjon tatt i bruk i stor stil (FFT), og når ble wavelettransformasjon tatt i bruk i betydelig grad?
5. Wavelettransformasjon har et "randproblem". Hva menes med det? Hvor stor er randverdisonen?
6. Kan du skissere hvordan wavelettransformasjon kan tenkes brukt for å generere noter direkte fra et lydopptak? Hvilke problemer ser du for deg kan forekomme?

Regneoppgaver

7. a) Bereng en Morlet wavelet (i tidsdomenet) for analysefrekvensen 250 og 750 Hz når samplingsfrekvensen er 5000 Hz og K-parameteren er 16. Plot resultatet med korrekte angivelser av tid på x-aksen.
   b) Beregn den fouriertransformerte av hver av de to waveletene. Bruk både en FFT direkte på Morlet-waveleten beskrevet i tidsbildet, og ved å beregne den fouriertransformerte direkte ved hjelp av ligning (14.10). Plot resultatene med korrekte angivelser av frekvens på x-aksen.
   c) Kontroller at toppunktet kommer på det stedet du skulle forvente.
   d) Gjenta punkt a-c også når K-parameteren er 50.
8. I denne oppgaven er det underliggende temaet analogien til Heisenbergs uskarphetsfunksjon.
   a) Generer en numerisk tallrekke som representerer et signal

$$f(t_n) = c_1 \sin(2\pi f_1 t_n) + c_2 \cos(2\pi f_2 t_n)$$

   hvor $n = 1, 2, ..., N$, $N = 8192$, $f_1 = 1000$ Hz, $f_2 = 1600$ Hz, $c_1 = 1.0$, $c_2 = 1.7$, og samplingsfrekvensen er 10 kHz. Signalet skal vare hele tiden vi betrakter signalet. Plot et passe utsnitt av signalet i "tidsbildet" (utslag som funksjon av tid) slik at detaljer kommer fram. Pass på å få korrekte tall samt tekst langs aksenene, gjerne også en overskrift.
   b) Beregn den fouriertransformerte av signalet. Plott et passe utsnitt av signalet i "frekvensbildet" (velg gjerne absoluttverdier av fourierkoeffisientene som funksjon av frekvens), med tall og tekst langs aksene som ovenfor.

c) Beregn den wavelettransformerte av signalet (kan godt ta utgangspunkt i den siste versjonen av programmet gitt i dette kapitlet, eller du kan skrive programmet mer eller mindre fra scratch selv). Bruk Morlet wavelets, og la analysefrekvensen gå f.eks. fra 800 til 2000 Hz (logaritmisk fordelt som vanlig innen wavelettransformasjon). Plot etter tur resultatet for bølgetallet K lik 24 og 200. Kommenter resultatet.
d) La så signalet være et harmonisk signal som før, men nå konvolutert med en gaussisk funksjon slik at vi får to "bølgepakker":

$$f(t_n) = c_1 \sin(2\pi f_1 t_n) \exp\left(-[(t_n - t_1)/\sigma_1]^2\right) + c_2 \cos(2\pi f_2 t_n) \exp\left(-[(t_n - t_2)/\sigma_2]^2\right)$$

hvor $t_1 = 0.15$ s, $t_2 = 0.5$ s, $\sigma_1 = 0.01$ s og $\sigma_2 = 0.10$ s. Beregn den fouriertransformerte av signalet, og også den wavelettransformerte av signalet. Plot signalet i tidsbildet, frekvensbildet (passe utsnitt) og den wavelettransformerte av signalet for K = 24 og 100 (test gjerne flere verdier!) og øvrige parametre som i deloppgave c). Kommenter resultatene!
e) I en lydfil med sangen fra en svarttrost som er tilgjengelig sammen med dette kapitlet, ber vi deg å bruke siste programsnutten i dette kapitlet (eller egen versjon) for å analysere en tidsstreng på vel 1.4 s. Parametre ved analysen: Filnavn: 'Svarttrost2.wav', Nstart = 17000, datastrengens lengde 64 k, frekvensområde 1500-8000 Hz, bølgetallet K lik 12 og 96 (og gjerne noen mellom disse verdiene også). Signalet består av fem ulike lydgrupper. Vi er først og fremst interessert i den fjerde av disse!

Plot signalet i tidsbildet, frekvensbildet og signalet analysert ved wavelettransformasjon for denne fjerde lydgruppen. Pass på å ta med også noen utsnitt av de opprinnelige plottene for å få fram detaljer. Dette gjelder i særdeleshet tidsbildet av den opprinnelige lyden! Forhåpentligvis vil du da gjenkjenne et signal vi har støtt på minst to ganger i tidligere kapitler. Du bør gjenkjenne hvordan vi kan lage et slikt signal matematisk.

Nøye analyse av den fjerde biten av lydsignalet i (1) tidsbildet og (2) etter wavelettransformasjon gjør det mulig å se hvordan en nær analogi til Heisenbergs uskarphetsrelasjon spiller inn på en praktfull måte! For å få fullt utbytte bør du hente ut tidsforskjeller og frekvensforskjeller i diagrammene og sammenholde disse med waveletens utbredelse i tids- og frekvensbildet for de to valgte K-verdiene.

Dersom du er student og har tilbud om hjelp fra lærere, anbefaler vi sterkt at du diskuterer de aktuelle detaljene med læreren inntil at du gjennomskuer samspillet vi ønsker å få fram. Det er mye verdifull kunnskap å hente fra dette problemet, kunnskap som også kan være verdifull i mange andre deler av fysikken!

9. Foreta en waveletanalyse av lydfilen "bokfink" på kursets websider. Forsøk en K-faktor som er dobbelt så stor som i eksemplet i figur 14.12. Forsøk også en analyse med halvparten av K-verdien som ble brukt i den nevnte figuren. Beskriv forskjellene du ser.

10. Foreta waveletanalysen av bokfinklyden på ny for $K = 48$. Velg etter tur å bruke "power spectrum" (absoluttverdien etter omvendt fouriertransformasjon kvadrert), waveletanalyse "på amplitudenivå" (absoluttverdien etter omvendt fouriertransformasjon direkte) og waveletanalyse med kvadratroten av waveletanalysen (kvadratroten av absoluttverdien etter omvendt fouriertransformasjon). Gjør dine vurderinger av hvilken av disse metodene du liker best for akkurat dette signalet. Kan det hende at du for et annet signal ville foretrukket en av de andre anskuelsesformene (kvadrat, rett fram, eller kvadratrot-fremstillingene)?

11. Bruk kunnskapen fra kapittel 1 og 2 til å beregne tidsforløpet for en fjærpendel etter at den er satt i sving av en harmonisk kraft med frekvens lik resonansfrekvensen. Følg svingningene også en tid etter at den harmoniske kraften er fjernet. Foreta så en waveletanalyse av svingeforløpet. Forsøk å optimalisere analysen mhp K-verdi. Finner du en tilsynelatende sammenheng mellom Q-verdien for pendelsvingningen og K-verdien som gir optimalt waveletdiagram?

12. Velg selv en lydfil som du kan transformere til en .wav-fil, og velg et utsnitt som du kan ha lyst til å analysere. Optimaliser analysen, og fortell hvilken informasjon du får ut av diagrammet.

13. Finn data på nettet som viser en tidssekvens du synes kan være interessant å studere. Det kan være værdata, solflekker, strømforbruk eller hva du måtte finne på. Analysér datasettet både ved tradisjonel fouriertransformasjon og med waveletanalyse. Hvilken metode synes du egner seg best for de dataene du valgte? (Bør ha data med en eller annen form for løs peridiositet med minst 20-30 perioder innenfor dataene du har tilgjengelig.)