Fourierrekker fra bilder

Av Benjamin Dybvik, Tobias Blegeberg og Ole Martin Ruud

Innholdsfortegnelse

1. Introduksjon

Hensikten med denne rapporten er å få en innføring i hva fourierrekker er, og hvorfor fourierrekker brukes i flere løsninger på praktiske anvendelser. For å forstå hvorfor fourierrekker brukes skal vi først se på anvendelsen av fourierrekker i dagligdagse objekter, og litt generelt om anvendelser. Vi skal ikke gå inn i matematikken bak disse anvendelsene, men hovedfokuset kommer til å ligge på hvordan fourierrekker kan løse utfordringene i anvendelsene. Hovedfokuset i rapporten er å få en innføring i fourierrekker og hvordan man framstiller et bilde med fourierrekker. Bakgrunnen for at vi har valgt framstilling av bilder er fordi det er en spennende anvendelse av fourierrekker. Å framstille bilder som fourierrekker er en metode som er mye brukt på bakgrunn av at man kan komprimere filstørrelsen på bildefilen betydelig, uten å miste for mye av oppfattet kvalitet. Dette er en praktisk tilnærming til fourierrekker og det er derfor lett å sette seg inn og forstå hensikten med fourierrekker.

2. Grunnleggende om fourierrekker

Introduksjon til fourierrekker

En fourierrekke er en sum av harmoniske funksjoner med ulik frekvens og amplitude som er tilnærmet lik en periodisk funksjon. Med uendelig antall ledd i rekken så er fourierrekken til en funksjon lik funksjonen, men med en endelig mengde ledd så er fourierrekken tilnærmet lik funksjonen.

Utregningen av fourierrekker kalles harmonisk analyse. Fourierrekken er svært nyttig, og kan brukes til å gjøre en tilfeldig periodisk funksjon om til en sum av enkle ledd. Man kan også

velge antall ledd man ønsker basert på hvor nøyaktig man ønsker å gjennomføre tilnærmingen.

Den generaliserte fourierrekken

Hvis vi bruker den generaliserte fourierrekken som benytter de ortogonale harmoniske funksjonene sinus og cosinus får vi at vi kan utlede fourierrekken til en funksjon ved hjelp av to koeffisienter, et konstant ledd og to summer gitt under. Hvis det er av interesse, så er utledningen av disse generaliserte formlene gjennomført av Eric W. Weisstein i sin artikkel om "Generalized Fourier Series".

Anta at $f(t)$ en tilfeldig periodisk funksjon med periode T og at c **in**~ Re = . Da får vi at fourierrekken til funksjonen $f(t)$ er $\psi(t)$. Under har vi også variablen ω som er vinkelhastigheten. Den er definert som $\omega = \frac{1}{T}$.

$$\psi(t) = a_0 + Sum(a_n \cdot \cos(n\omega t) + b_n \cdot \sin(n\omega t), n = 1 \ldots \text{infinity})$$

$$\psi(t) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos(n\omega t) + b_n \sin(n\omega t)\right) \quad \textbf{(2.2.1)}$$

$$a_0 = \frac{2}{T} \cdot Int(f(t), t = c \ldots (c+T))$$

$$a_0 = \frac{2\left(\int_c^{c+T} f(t)\, dt\right)}{T} \quad \textbf{(2.2.2)}$$

$$a_n = \frac{2}{T} \cdot Int(f(t) \cdot \cos(n\omega t), t = c \ldots (c+T))$$

$$a_n = \frac{2\left(\int_c^{c+T} f(t)\cos(n\omega t)\, dt\right)}{T} \quad \textbf{(2.2.3)}$$

$$b_n = \frac{2}{T} Int(f(t) \cdot \sin(n\omega t), t = c \ldots (c+T))$$

$$b_n = \frac{2\left(\int_c^{c+T} f(t)\sin(n\omega t)\, dt\right)}{T} \quad \textbf{(2.2.4)}$$

Hvis $f(t)$ har diskontinuiteter, er fourierrekken til $f(t)$

$$\psi(t) = \frac{f(t^+) + f(t^-)}{2}$$

$$\psi(t) = \frac{f(t)}{2} + \frac{f(t^-)}{2} \quad \textbf{(2.2.5)}$$

Dette vil også være sant t **in**~ Re = $t \in \Re$ når funksjonen er kontinuerlig, men da vi uttrykket kunne bli faktorisert til $\psi(t) = f(t)$. Vi sier at $f(t) \sim \psi(t)$.

Odde og like funksjoner

Ved gitte vilkår kan det gjøres noen forenklinger av formelen for fourierrekken. Disse vil vise seg å være nyttige når man skal gjøre et bilde om til en fourierrekke.

Odde funksjoner

Når vi finner fourierrekken til en odde funksjon finner vi ut at vi mister de like leddene i uttrykket. $f(t)$ er en odde funksjon og $\cos(n\omega t)$ er en like funksjon, derfor må produktet $f(t)\cdot\cos(n\omega t)$ være en odde funksjon. Integralet over en periode til en odde funksjon er 0.

Anta at c **in**~Re = $c \in \Re$ og at funksjon av $f(t)$ er en periodisk, odde funksjon med periode T. ta at f(t) en tilfeldig periodisk funksjon med periode T og at c **in**~ Re = $c \in \Re$. Da får vi at fourierrekken til funksjonen $f(t)$ er $\psi(t)$. Under har vi også variablen ω som er vinkelhastigheten. Den er definert som $\omega = \frac{1}{T}$.

$a_n = \frac{2}{T}\cdot Int(f(t)\cdot\cos(n\omega t), t = c\ldots(c+T))$;
$a_n = 0$

$$a_n = \frac{2\left(\int_c^{c+T} f(t)\,\cos(n\omega t)\,dt\right)}{T}$$

$$a_n = 0 \qquad \textbf{(2.3.1)}$$

Uttrykket for fourierrekken til en odde funksjon blir derfor:

$\psi(t) = Sum(b_n\cdot\sin(n\omega t), n = 1\ldots\text{infinity})$

$$\psi(t) = \sum_{n=1}^{\infty} b_n \sin(n\omega t) \qquad \textbf{(2.3.2)}$$

I tillegg kan vi forenkle uttrykket av b_n slik at vi bare trenger å integrere over en halv periode.

$b_n = \frac{4}{T}\cdot Int\left(f(t)\cdot\sin(n\omega t), t = c\ldots\left(c+\frac{T}{2}\right)\right)$

$$b_n = \frac{4\left(\int_c^{c+\frac{T}{2}} f(t)\,\sin(n\omega t)\,dt\right)}{T} \qquad \textbf{(2.3.3)}$$

Like funksjoner

Når $f(t)$ derimot er like, så skjer det mottsatte. Nå er det derimot når vi ganger med $\sin(n\omega t)$ at integralet blir 0. Det er fordi nå er $f(t)$ en like funksjon og $\sin(n\omega t)$ er en odde funksjon. Produktet av disse blir derfor en odde funksjon, og integralet over en periode til en odde funksjon er 0.

Anta at c **in**~Re = $c \in \Re$ og at funksjon av $f(t)$ er en periodisk, like funksjon med periode T.

$b_n = Int(f(t)\cdot\sin(n\omega t), t = c\ldots(c+T))$;
$b_n = 0$

$$b_n = \int_c^{c+T} f(t)\ \sin(n\omega t)\ \mathrm{d}t$$

$$b_n = 0 \qquad \textbf{(2.3.4)}$$

Dermed blir uttrykket for fourierrekken til en like funksjon:

$$\psi(t) = \frac{a_0}{2} + Sum\left(a_n \cdot \cos(n\omega t), n = 1 \ldots \text{infinity}\right)$$

$$\psi(t) = \frac{a_0}{2} + \left(\sum_{n=1}^{\infty} a_n \cos(n\omega t)\right) \qquad \textbf{(2.3.5)}$$

Her kan vi også forenkle uttrykket for a_0 og a_n slik at vi bare trenger å integrere over en halv periode.

$$a_0 = \frac{4}{T} \cdot Int\left(f(t), t = c \ldots \left(c + \frac{T}{2}\right)\right);$$
$$a_n = \frac{4}{T} \cdot Int\left(f(t) \cdot \cos(n\omega t), t = c \ldots \left(c + \frac{T}{2}\right)\right)$$

$$a_0 = \frac{4\left(\int_c^{c+\frac{T}{2}} f(t)\ \mathrm{d}t\right)}{T}$$

$$a_n = \frac{4\left(\int_c^{c+\frac{T}{2}} f(t)\ \cos(n\omega t)\ \mathrm{d}t\right)}{T} \qquad \textbf{(2.3.6)}$$

Den komplekse fourierrekken

Fourierrekker består av to harmoniske funksjoner, sinus og cosinus. Ved hjelp av Eulers formel kan vi gjøre disse om til et utrykk av e. Eulers formel forteller oss at

$$\cos(x) = \frac{e^{ix} + e^{-ix}}{2};\ \sin(x) = \frac{e^{ix} - e^{-ix}}{2\,i};$$

$$\cos(x) = \frac{e^{ix}}{2} + \frac{e^{-ix}}{2} \qquad \textbf{(2.4.1)}$$

$$\sin(x) = \frac{e^{ix} - e^{-ix}}{2\,i} \qquad \textbf{(2.4.1)}$$

Når vi setter (2.3.1) inn i formelen (2.2.1) for fourierrekken, kan vi utlede den komplekse formen av fourierrekken.

$$\psi(t) = \frac{a_0}{2} + Sum\left(a_n \cdot \cos(n\omega t) + b_n \cdot \sin(n\omega t), n = 1 \ldots \text{infinity}\right)$$

$$\psi(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(n\omega t) + b_n \sin(n\omega t) \right) \tag{2.4.2}$$

$$\psi(t) = \frac{a_0}{2} + Sum\left(a_n \cdot \frac{e^{i\cdot n\cdot\omega} + e^{-i\cdot n\cdot\omega}}{2} + b_n \cdot \frac{e^{i\cdot n\cdot\omega} - e^{-i\cdot n\cdot\omega}}{2\cdot i}, n = 1 \ldots \text{infinity} \right)$$

$$\psi(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \left(\frac{e^{i\,n\,\omega}}{2} + \frac{e^{-i\,n\,\omega}}{2} \right) + \frac{b_n \left(e^{i\,n\,\omega} - e^{-i\,n\,\omega} \right)}{2\,i} \right) \tag{2.4.3}$$

$$\psi(t) = \frac{a_0}{2} + Sum\left(\frac{a_n - i\cdot b_n}{2} \cdot e^{i\cdot n\cdot\omega\cdot t}, n = 1 \ldots \text{infinity} \right) + Sum\left(\frac{a_n + i\cdot b_n}{2} \cdot e^{-i\cdot n\cdot\omega\cdot t}, n = 1 \ldots \text{infinity} \right)$$

$$\psi(t) = \frac{a_0}{2} + \left(\sum_{n=1}^{\infty} \frac{a_n - i\,b_n\,e^{i\,n\,\omega\,t}}{2} \right) + \left(\sum_{n=1}^{\infty} \left(\frac{i\,b_n}{2} + \frac{a_n}{2} \right) e^{-i\,n\,\omega\,t} \right) \tag{2.4.4}$$

Får å kunne skrive fourierrekken på en enkel måte så gjør vi noen definisjoner.

$$c_0 = \frac{a_0}{2}$$

$$c_0 = \frac{a_0}{2} \tag{2.4.5}$$

$$c_n = \frac{a_n - i\cdot b_n}{2}$$

$$c_n = -\frac{i\,b_n}{2} + \frac{a_n}{2} \tag{2.4.6}$$

$$c_{-n} = \frac{a_n + i\cdot b_n}{2}$$

$$c_{-n} = \frac{i\,b_n}{2} + \frac{a_n}{2} \tag{2.4.7}$$

Deretter setter vi disse definisjonene inn i uttrykket (2.3.4) ovenfor så får vi som vi utledet over.

$$\psi(t) = \frac{a_0}{2} + Sum\left(\frac{a_n - i\cdot b_n}{2} \cdot e^{i\cdot n\cdot\omega\cdot t}, n = 1 \ldots \text{infinity} \right) + Sum\left(\frac{a_n + i\cdot b_n}{2} \cdot e^{-i\cdot n\cdot\omega\cdot t}, n = 1 \ldots \text{infinity} \right)$$

$$\psi(t) = \frac{a_0}{2} + \left(\sum_{n=1}^{\infty} \frac{a_n - i\,b_n\,e^{i\,n\,\omega\,t}}{2} \right) + \left(\sum_{n=1}^{\infty} \left(\frac{i\,b_n}{2} + \frac{a_n}{2} \right) e^{-i\,n\,\omega\,t} \right) \tag{2.4.8}$$

$$\psi(t) = c_0 + Sum\left(c_n \cdot e^{i\cdot n\cdot\omega\cdot t}, n = 1 \ldots \text{infinity} \right) + Sum\left(c_{-n} \cdot e^{-i\cdot n\cdot\omega\cdot t}, n = 1 \ldots \text{infinity} \right)$$

$$\psi(t) = \frac{a_0}{2} + \left(\sum_{n=1}^{\infty} c_n e^{i n \omega t} \right) + \left(\sum_{n=1}^{\infty} c_{-n} e^{-i n \omega t} \right) \quad \textbf{(2.4.9)}$$

$c_n = \frac{1}{T} Int\left(f(t) \cdot e^{-i \cdot n \cdot \omega \cdot t}, t = c \ldots (c+T) \right)$

$$c_n = \frac{\int_c^{c+T} f(t)\, e^{-i n \omega t}\, dt}{T} \quad \textbf{(2.4.10)}$$

Resultatet (2.3.9) fra denne utledningen kalles den komplekse formen av fourierrekken. Den komplekse formen for fourierrekker er algebraisk enklere og mer symmetrisk, derfor brukes denne formen av fourierrekker ofte i blant annet fysikk. For å komme rett fra funksjonen til den komplekse formen av fourierrekken kan vi bruke (2.3.10).

På den komplekse formen av fourierrekken så er det et teorem som heter Parsevals Teorem som er greit å ha sett. Hvis vi antar at $f(t)$ er en periodisk funksjon med en periode på T og at c **in**~ Re = $c \in \Re$.

$\frac{1}{T} Int\left((|f(t)|)^2, t = c \ldots (c+T) \right) = Sum\left((|C_n|)^2, n = -\text{infinity} \ldots \text{infinity} \right)$

$$\frac{\int_c^{c+T} |f(t)|^2\, dt}{T} = \sum_{n=-\infty}^{\infty} |C_n|^2 \quad \textbf{(2.4.11)}$$

Hvis $f(t)$ er reell, altså at det ikke har en imaginær del, da får vi videre at.

$Sum\left((|C_n|)^2, n = -\text{infinity} \ldots \text{infinity} \right) = (|C_0|)^2 + 2 \cdot Sum\left((|C_n|)^2, n = 1 \ldots \text{infinity} \right)$

$$\sum_{n=-\infty}^{\infty} |C_n|^2 = |C_0|^2 + 2 \left(\sum_{n=1}^{\infty} |C_n|^2 \right) \quad \textbf{(2.4.12)}$$

Et eksempel på en fourierrekke

Under skal vi vise fra et eksempel på en periodisk funksjon $f(t)$ som vi skal lage en fourierrekker til. T er perioden til funksjonen.

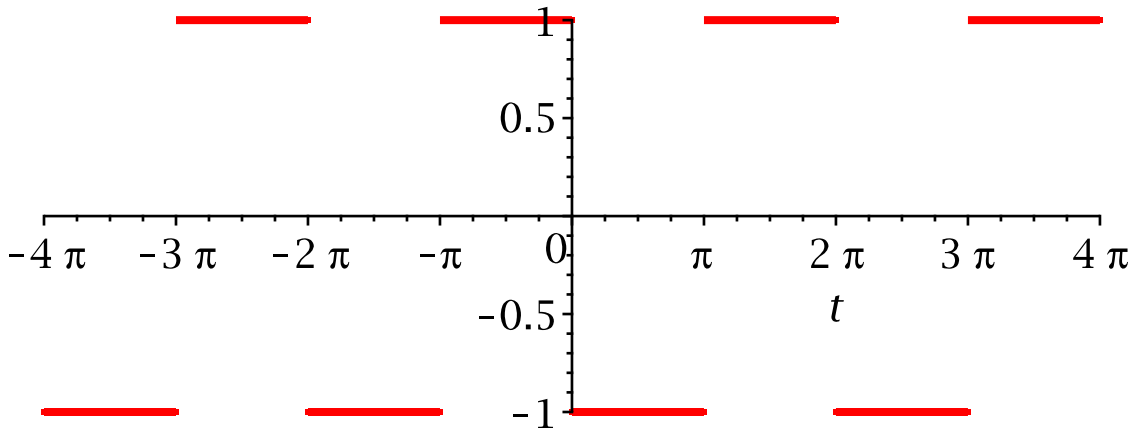
$f := t \to piecewise(0 \le t \le \text{Pi}, -1, \text{Pi} < t \le 2 \cdot \text{Pi}, 1)$; $T := 2 \cdot \text{Pi}$;

$$f := t \mapsto \begin{cases} -1 & 0 \le t \le \pi \\ 1 & \pi < t \le 2\pi \end{cases}$$

$$T := 2\pi \quad \textbf{(2.5.1)}$$

$P_s := j \to plot(f(t - j \cdot T), t = j \cdot T \,..\, T \cdot (j+1), discont = true, symbol = \text{"point"}, color = \text{"red"}, thickness = 3)$:

$P_f := plots\text{:-}display\left([seq(P_s(j), j = -2 \,..\, 1)] \right)$



$c_n := n \to int\left(f(t) \cdot \exp\left(-I \cdot n \cdot \left(\frac{2 \cdot Pi}{T} \right) \cdot t \right), t = 0 \ldots T \right)$

$$c_n := n \mapsto \int_0^T f(t)\ e^{\frac{-2\,I\,n\,\pi\,t}{T}}\ dt \qquad \textbf{(2.5.2)}$$

$\psi := (t, I) \to sum\left(\frac{1}{T} \cdot c_n(n) \cdot \exp\left(I \cdot n \cdot \left(\frac{2 \cdot Pi}{T} \right) \cdot t \right), n = -I \ldots I \right)$

$$\psi := (t, I) \mapsto \sum_{n=-I}^{I} \frac{c_n(n)\ e^{\frac{2\,I\,n\,\pi\,t}{T}}}{T} \qquad \textbf{(2.5.3)}$$

Her regner vi ut fourierrekken til $f(t)$ (2.4.1) med I antall ledd.

$I := 7$:

$\psi_1 := evalc(\psi(t, I))$

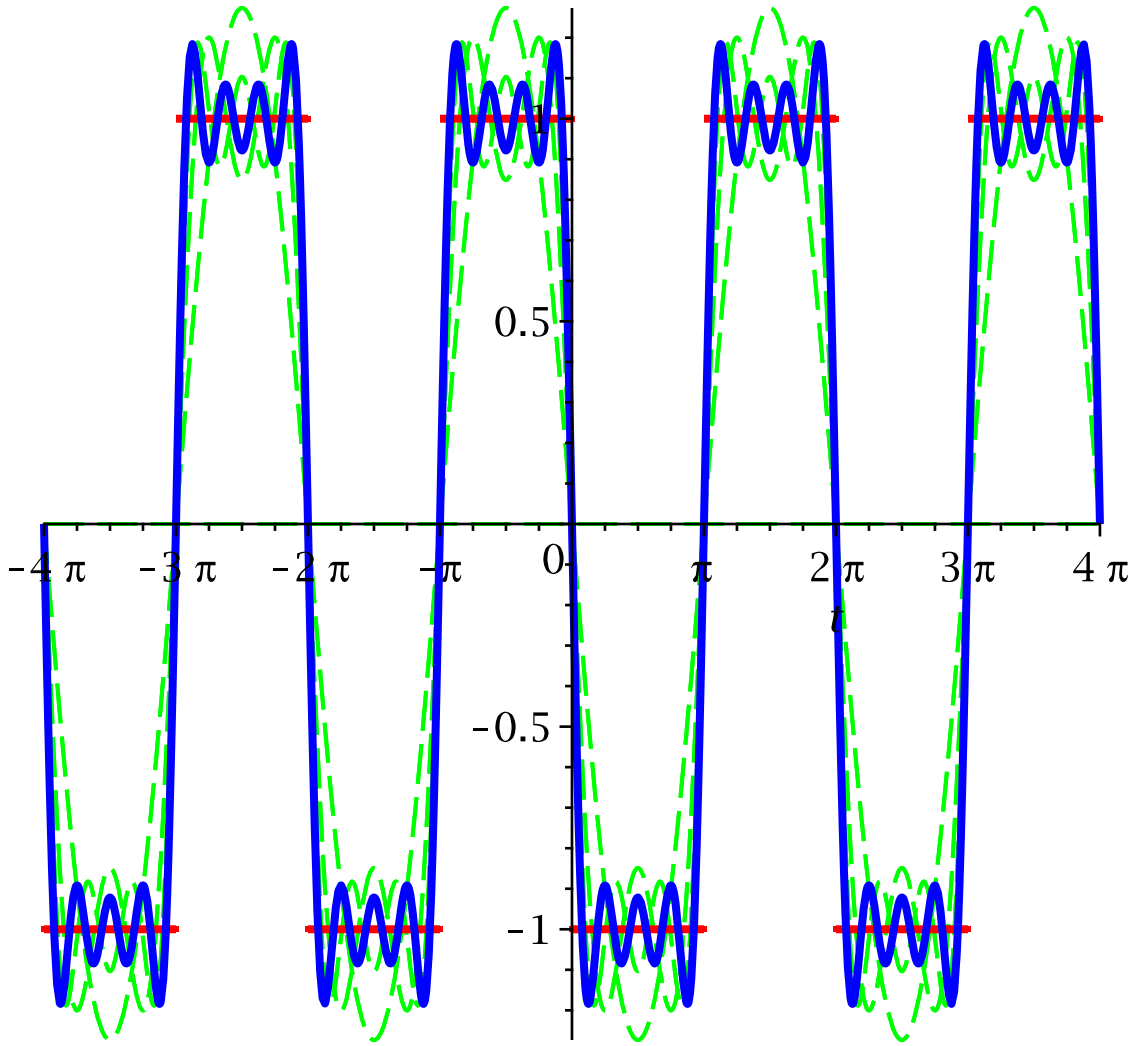
$$\psi_1 := -\frac{4 \sin(7\,t)}{7\,\pi} - \frac{4 \sin(5\,t)}{5\,\pi} - \frac{4 \sin(3\,t)}{3\,\pi} - \frac{4 \sin(t)}{\pi} \qquad \textbf{(2.5.4)}$$

Her plottes fourierrekken (2.4.4), vist her i blått, med I antall ledd og orginalfunksjonen $f(t)$ (2.4.1), vist her i rødt. I tillegg er det tegnet opp hver fourierrekke med færre ledd enn rekken (2.4.4) vi regnet ut. Slik kan vi se hvordan flere ledd endrer på rekke. Rekkene med færre ledd I er grønne.

$P_\psi := plot\left(\psi_1, t = -2 \cdot T \ldots 2 \cdot T, color = \text{"blue"}, thickness = 3 \right)$:

$Pr_\psi := plot\left(\left[seq\left(\psi(t, j), j = 0 \ldots (I - 1) \right) \right], t = -2 \cdot T \ldots 2 \cdot T, color = \text{"green"}, linestyle = \text{"dash"} \right)$:

$plots{:}\text{-}display\left(\left[Pr_\psi, P_f, P_\psi \right] \right)$



3. Noen anvendelser av fourierrekker

<Midlertidig Overskrift>

Det viser seg at man kan bruke fourierrekker til å framstilling nesten alle slags bølger som en funksjon. Som vist tidligere kan en fourrierrekke kunne tilnærme seg en bølge ved å ha uendelig antall cosinus og sinus ledd. Dette gir oss en enorm mulighet til å framstille mange ting rundt oss som funksjoner ettersom så mye vi omgår dagligdags enten er eller kan oppfattes som bølger. Eksempler på slike dagligdagse bølger vi omgår kan være i det elektromagnetiske spekteret slik som signaler fra en mobil eller lydspekteret slik som musikk eller en stemme. Mekaniske ting som går i en viss takt kan vi også framstille som cosinus og sinus bølger. Eksempler på dette er klokker, motorer eller kraftverk der maskinene går i en fast syklus. [1] Ingeniører bruker fourerrekker til å illustrere forskjellige bølger som funksjoner. Oppfinneren av Fourierrekker Joseph Fourier brukte fourierrekker for å regne på varmetap, varmebølger og vibrasjoner.

Ingeniører bruker fourerrekker innen mange fagfelt. For å kunne regne på høy og lavvann og hvordan bølger påvirker et skip, til komprimering av data og tolkning av elektromagnetiske bølger slik som i telekommunikasjon. Dette gjelder mange områder innenfor signalbehandling, slik som radar, sonar, x-ray, mobiltelefoni og samband. I for eksempel Wifi brukes fourerrekker for å kode og dekode signalet presist slik at det kan bli tolket digitalt. Det

brukes også for å restaurere signaler som er i ferd med å «viskes ut» på grunn av dårlig dekning. [3]

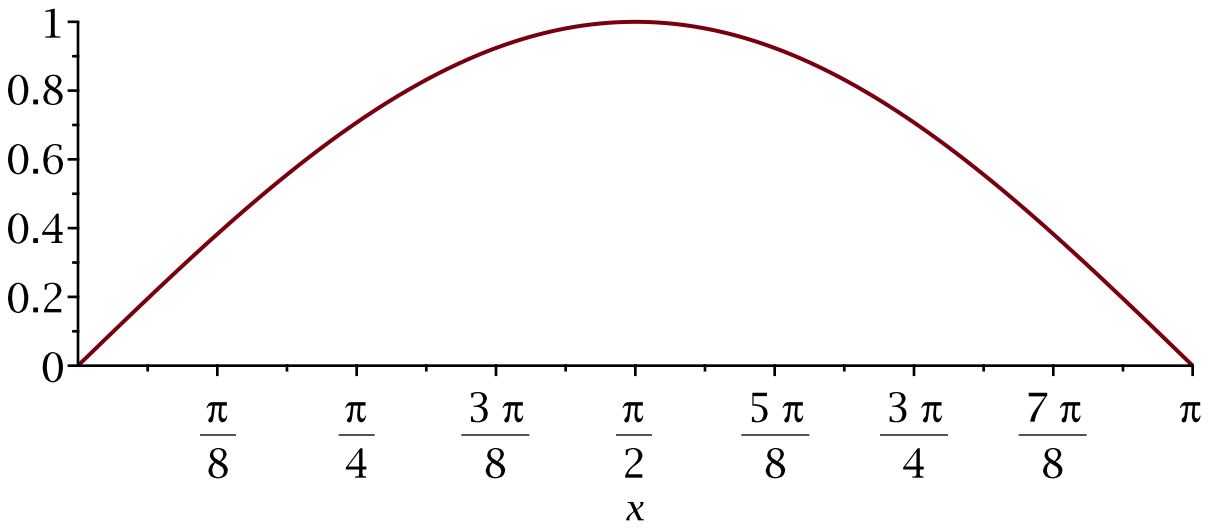
Ved å ta opp lyd kan man konstruere dette om til en fourerrekke funksjon. Denne funksjonen består av mange ledd som går an å dele opp og analysere separat. Ved å analysere de forskjellige frekvensene som den originale lyden består av, kan man danne seg en graf. Et program som Shazam bruker denne teknikken til å kunne finne ut hvilken sang man hører på. Shazam tar grafen og sammenligner den med en database for å se om den matcher. Om Shazam finner en match vil den fortelle deg hvilke sang du hører på. [2] Smarttelefon stemmegjenkjenningsprogramvare slik som Siri bruker samme teknologi for å forstå hva du ber den om å gjøre. Ved å analysere stemmen din og sjekke det opp med en Apple database skal den være i stand til å gjøre som du ber den om. Så langt kan man få et lite inntrykk av hva man kan framstille med fourerrekker og det er langt flere muligheter. Fourerrekker kan være et veldig kraftig verktøy i forskjellige anvendelser om det blir brukt smart.

[1] https://math.stackexchange.com/questions/579453/real-world-application-of-fourier-series 21.09.17
[2] https://www.youtube.com/watch?v=K0_TP5Sf7yc 21.09.17
[3] https://www.quora.com/Why-are-Fourier-series-important-Are-there-any-real-life-applications-of-Fourier-series 21.09.17

Eksempel på anvendelse i musikkinstrumenter

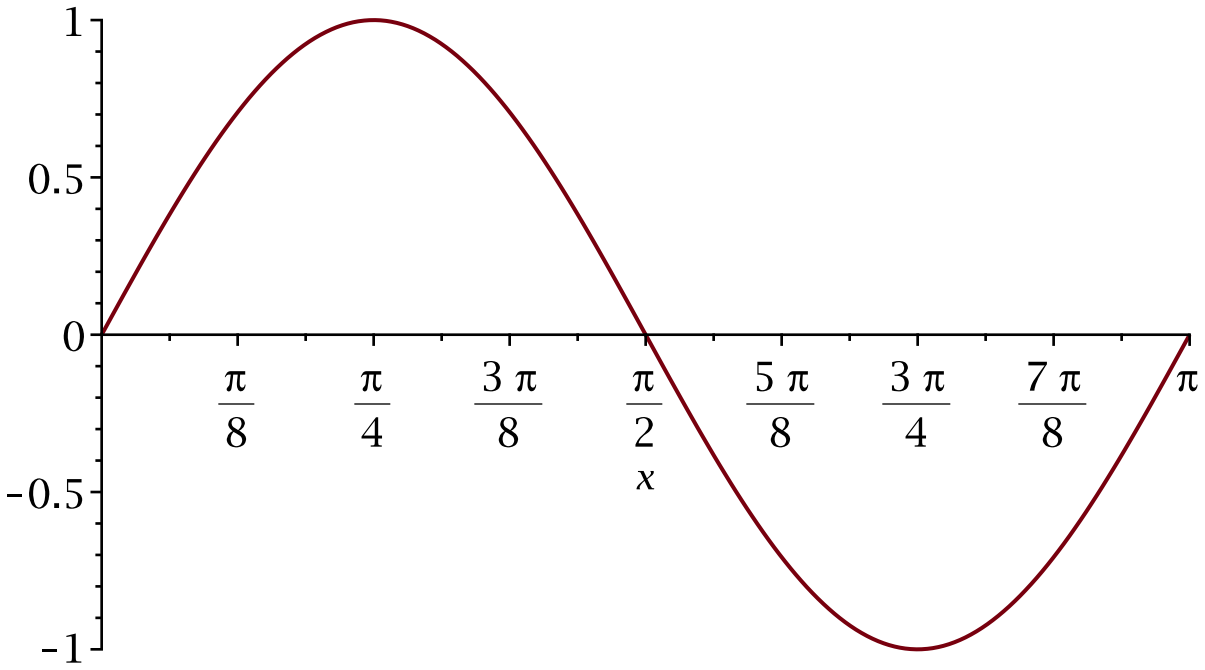
La oss se litt nærmere på hvordan Fourierrekker blir anvendt i dagligdagse ting og objekter, som for eksempel lyd og instrumenter. Hva er det som skjer når man drar buen over en cello? Før må vi finne ut hva lyd er. En simpel forklaring er at lyd er fortynninger og fortetninger i luften, eller et annet medium som lydbølgene skulle bevege seg i. Det er trommehinnen som greier å registrer endringene i lufttrykket, fordi vi har membraner inne i øret som vil svinge i samme takt som lydbølgene. Grunnen til at celloen kan lage disse harmonisk lydbølge med riktig frekvens, er på grunn av celloens utforming.

Celloen har flere strenger med en bestemt lengde som er tilpasset tonen den skal lage, i dette eksemplet bestemmer vi at lengden på cellostrengen er π. Vi legger cellostrengen langs x-aksen, og siden lengden på strengen er π blir også perioden $T = \pi$. Av disse opplysningene kan vi lage en funksjon av cellostrengen, nemlig $f(x) = sin(x)$ illustrert under.



Figur x

Dette er utslaget man får hvis man drar buen over celloen, vi definere dette som en tone på 65 hz (dette er en C), altså at grafen svinger opp og ned 65 ganger i sekundet. Så legger man en finger i punktet $\frac{\pi}{2}$, da får vi funksjonen $f(x) = sin(2x)$.



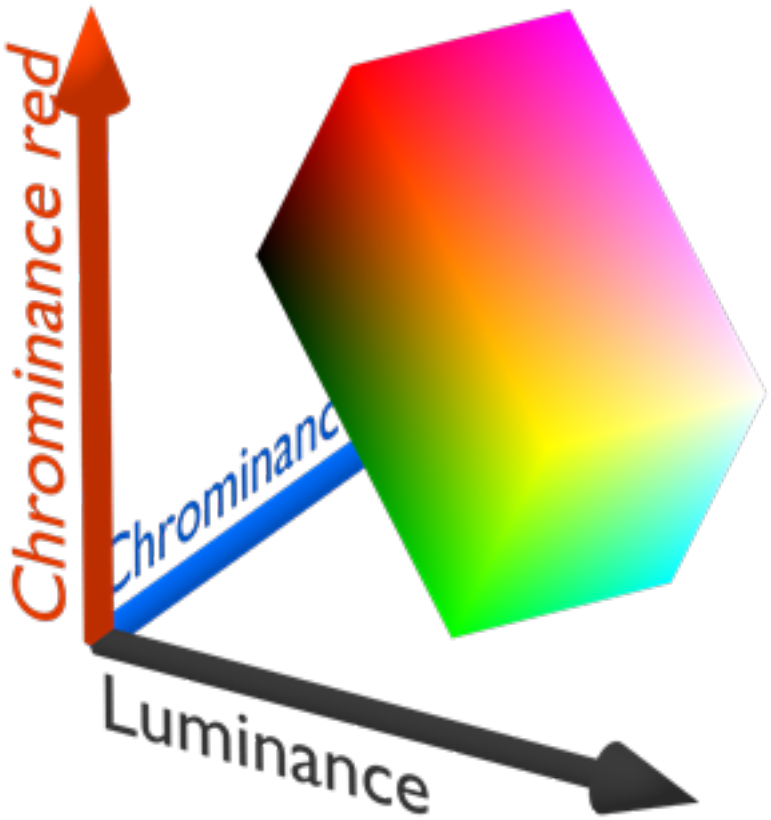
Dette er fouriersinusrekker som kan utledes videre på formen $f(x) = \sum_{i=1}^{\infty} b_n \cdot sin(n\varpi x)$. Dette er en tone på 130 hz (også en C, men høyere), det vi si dobbelt så mange svingninger. Vi skal ikke utlede fourierrekken nå, men vite at for hver gang frekvensen dobler seg, får man den sammen tronen, men i en skala opp. Dette er mye brukt innenfor for eksempel musikkteori.

4. Lage en fourierrekke fra et bilde

Hvordan fungerer JPEG

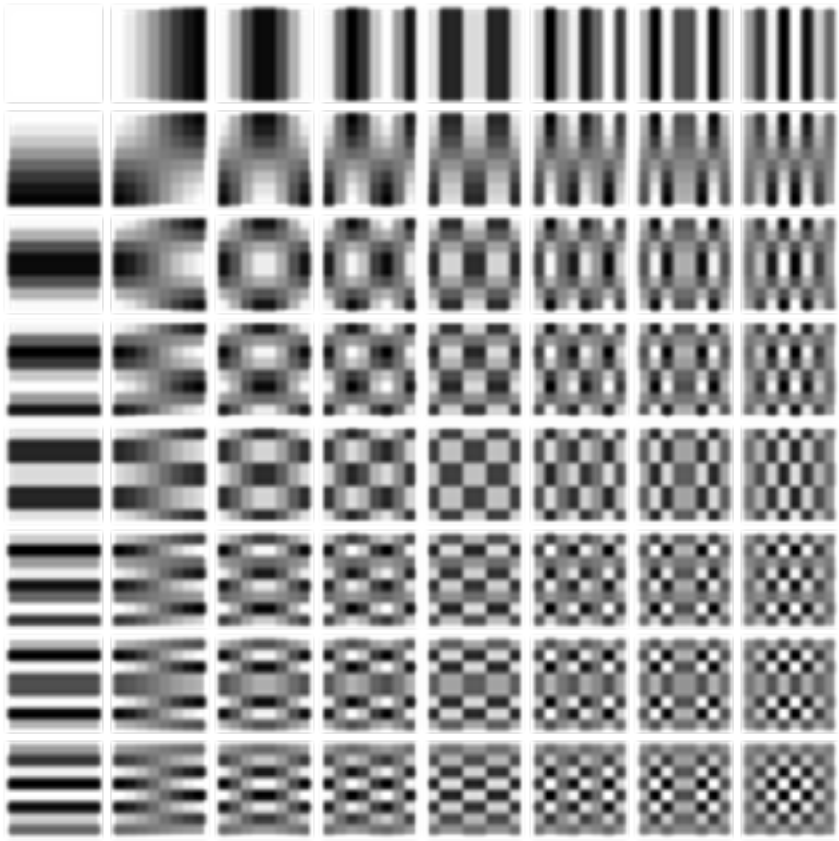
JPEG (Joint Photographic Experts Group) er et filformat for lagring av biler. Mens vanlig bildeformat lagrer en verdi for hvert piksel så tar JPEG å komprimerer bildet. Dette resulterer at det tar opp mindre plass enn originalfilen. Det finnes forskjellige versjoner av JPEG. Vi kan få JPEG der bildekvaliteten ikke blir tapt og JPEG der bildekvalitet blir tapt. I utgangspunktet er det ønskelig å bruke JPEG med tap av bildekvalitet. Det er hensiktsmessig fordi litt tap i bildekvalitet kan føre til at man får komprimert bildet betydelig bedre. Den dataen som blir luket ut av bildet har lite eller ikke noe å si på hvordan vi oppfatter slutt produktet.

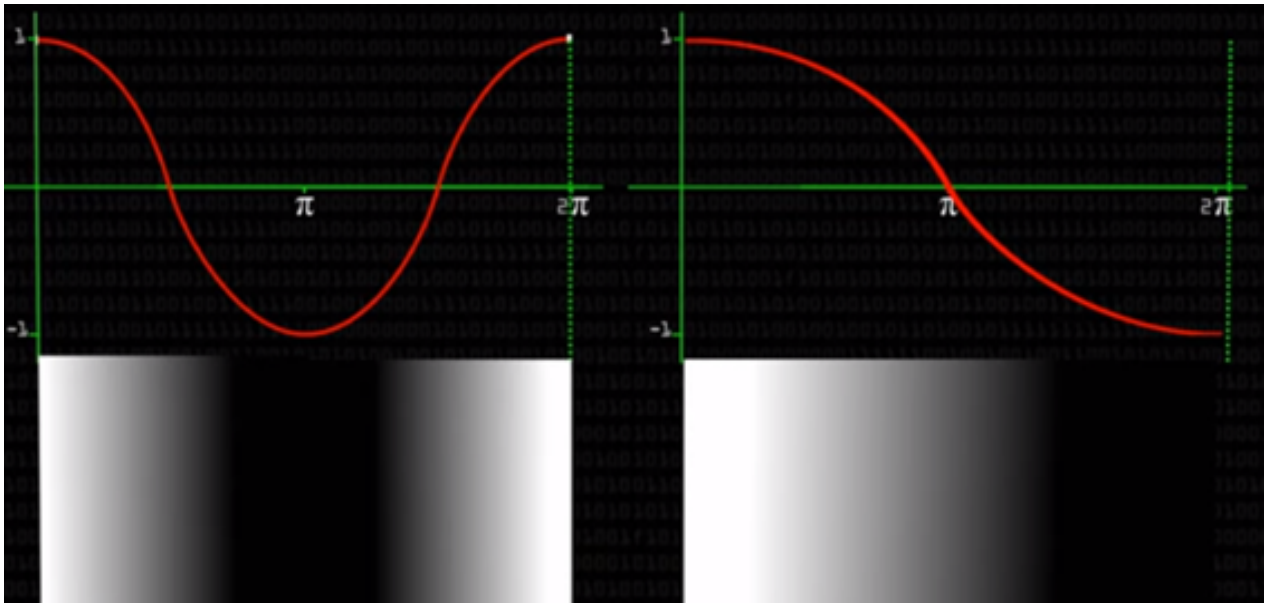
Først blir pikslene gjort om fra RGB til noe som heter YCbCr. YCbCr er en fargemodell slik som RGB. Istedenfor å representere fargen gjennom en fargekombinasjon av rød, grønn og blå så kan YCbCr representere det samme fargespekteret gjennom Y, Cb og Cr. Y er luminans, det er et mål på hvor lyst et område er. Cb og Cr er rødnyanser og blånyanser til bildet. Gjennom disse tre komponentene får vi det samme fargespekteret som RGB gir oss.



Kilde: https://en.wikipedia.org/wiki/YCbCr hentet: 02.10.17

Grunnen til at vi gjør om fra RGB til YCrCb er fordi at vi kan redusere data fra den sistnevnte uten at det endrer hvordan vi oppfatter det ferdige resultatet. Øynene våre er relativt dårlige på å se forskjell mellom fargenyanser. Dette gjør av vi kan kutte ut en del data fra Cb og Cr uten at det reduserer vårt inntrykk av hvordan vi oppfatter bildet. Ved å fjerne data fra Cr og Cb kutter vi litt av kvaliteten på bildet. Dette er noe av bildekvalitetstapet som vi har med JPEG. Luminans oppfattes av oss mennesker som gråtoner. Vi mennesker mye bedre på å se nyanser av gråtoner enn nyanser av farger. Det blir som å se midt på natta når det er mørkt. Du kan skimte hva som er foran deg, men du klarer ikke å se farger. I YCrCb får vi avgrenset luminanskomponenten, slik at vi får redusert de andre komponentene uten å påvirke luminansen i bildet.
Deretter deler JPEG bildet opp i kvadrater på 8x8 piksler der hver fargekomponent er adskilt. Hver av disse komponentene som er delt opp i 8x8 blokker kan bli representert av Discrete Cosine Transformation (DCT), Diskré Cosinus Transformasjon. Diskré Cosinus Transformasjon er basert på Joseph Fourier sin Fourierrekke. I forskjell fra Fourierrekke bruker DCT kun cosinusledd, cosinus dette er for å slippe å ta stilling til både sinus og cosinus ledd og dermed redusere den totale dataen enda mer. Vi kan se på DCT som følgende matrise (til venstre);





Kilde: https://youtu.be/Q2aEzeMDHMA?t=4m35s 02.10.17

Denne matrisen er bygd opp av cosinusledd med forskjellige frekvenser i x planet (horisontalt) og y planet (vertikalt).

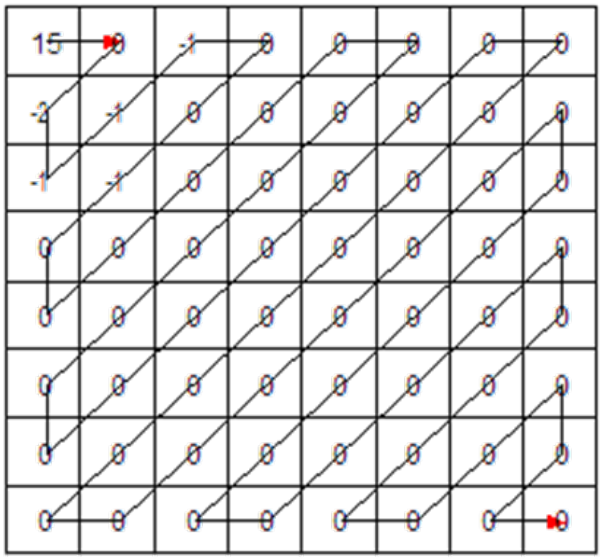
$a_n \cos(n\omega t)$

På bildet til høyre ser vi hvordan cosinusleddene representerer matrisen. Amplituden bestemmer nyansen av komplimentene til Y, Cr og Cb. Det er her snakk om komplimentfargene, vi kan se på bildet her at det er luminanskomponenten y. Frekvensen bestemmer hvor ofte det skal byttes mellom komplimentene. Matrisen oppe til venstre

kombinerer cosinusleddene i y planet og x planet og gis oss 8 ganger 8 antall kombinasjoner. Disse 64 kombinasjonene legges lagvis for å rekonstruere en 8x8 blokk med bilde. De ulike kombinasjonene trengs i ulik grad for å gjenskape bildet presist. For å bestemme hvor mye hver kombinasjon skal bidra regnes bidraget ut som koeffisienten av fourierrekken i DCT. DCT matrisen og vektleggingen av kombinasjonene er kun basert på Fourierrekker. Det gjør ved at fourierrekkeformelen $\psi(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(n\omega t) + b_n \sin(n\omega t) \right)$ er alt vi trenger for å gjenskape 8x8 blokken.

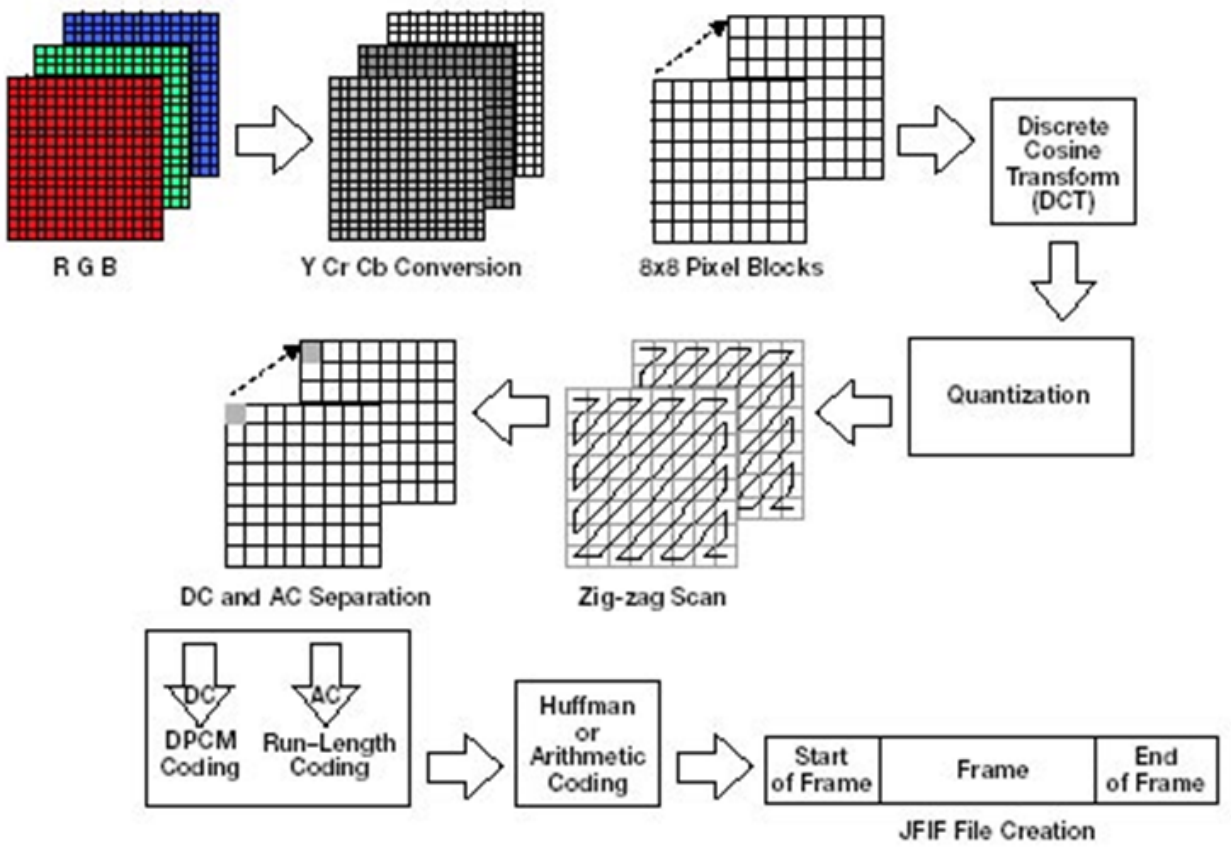
Videre for å redusere data betraktelig brukes en tabell som heter «Quantized table». Hensikten med kvantisering (quantized) er å fjerne de bidragene vi ikke vil ha, slik som de med veldig høy frekvens og de som har liten betydning for bildet. Dette gjøres fordi vi kan redusere filstørrelsen ved å utelukke disse leddene med uvesentlige detaljer. I tillegg er disse mindre detaljene i bildet vanskelig for mennesker å oppfatte, vårt helhetsinntrykk av bildet vil forbli det samme. En slik kvantiseringstabell er forhåndsbestemt av JPEG og inngår i et regnestykke sammen med koeffisienten som divisor og dividend for å bestemme hvor mye vi velger å vektlegge bidraget. En slik kvantisert tabell ser vi på bildet under til venstre. Legg merke til at verdiene nede til høyre er større enn de oppe til vestre. Dette er fordi det er ønskelig at verdiene nede til høyre veier mindre enn de verdiene oppe til venstre.
Koeffisient / kvantisert = ny data.
Bidragene etter kalkulasjonen vil være mindre enn de var før. I tillegg vil flesteparten av bidragene ha blitt desimaltall. Disse desimaltallene rundes av til nærmeste heltall. Siden en del av disse bidragene har blitt veldig små så vil de rundes av til null. Det vil bli større representasjon av null nede til høyre enn oppe til venstre ettersom de mindre bidragene i matrisen allerede har blitt dividert med de større kvantiseringsverdiene. Når vi da leser igjennom tabellen sikk-sakk vil de tellende verdiene komme først og etterfulgt av en lang rekke med null. Det er hensiktsmessig å lese igjennom tabellen sikk-sakk ettersom vi får plassert hele rekka med null bakerst og ved siden av hverandre. En slik lang rekke med null kan lett bli komprimert av Huffman koding. Bildet under til høyre viser hvordan dataen til bli lest om til en rekke.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99



Kilde: http://www.ece.ucdavis.edu/cerl/reliablejpeg/compression/ 02.10.17

Prosessen som vi nå har snakket om vil gjenta seg for alle 8x8 blokkene i bildet til bildet har blitt komprimert. Denne dataen vil da bli lagret som en komprimert JPEG fil. Hele prosessen er kort illustrert i følgende bilde



Kilde: http://www.eetimes.com/document.asp?doc_id=1225736 1.10.17

For å gjøre denne JPEG filen om til et forståelig bilde gjøres denne prosessen baklengs. Dataen blir dekomprimert og legges tilbake i en matrise. Deretter blir alle verdiene multiplisert med kvantiseringstabellen. Ettersom bidragene tidligere hadde blitt rundet av etter divisjonen så vil tallene nå bli noe annerledes. Til tross for det er verdiene i helhet ganske like og vil ikke påvirke vår oppfatning. Flesteparten av verdiene hadde blitt rundet ned til null. Disse verdiene vil forbli null etter at bidragene har blitt mulitplisert med kvantiseringstabellen. Disse verdiene gjelder i hovedsak for høye frekvenser, slike frekvenser som vi ikke oppfatter mye av, disse annuleringene vil dermed ikke få stor betydning for bildet heller. Når det er sagt så vil sluttproduktet være annerledes. Feil og mangler kan bli sett av et trent øye. Bidragene i forskjelllige frekvensområder blir lagt sammen for å gjenskape den originale 8x8 blokken. Blokkene blir så lagt side om side for å gjenskape det fulle bildet.

For å sammenligne JPEG og rådata er det her hentet ut noen sammenligninger fra "Image Compression and the Discrete Cosine Transform" av Ken Cabeen og Peter Gent fra College of the Redwoods.

Figure 5 – Original Peppers

Figure 6 – Quality 50 – 84% Zeros

Figure 7 – Quality 20 – 91% Zeros

Figure 8 – Quality 10 – 94% Zeros

Ovenfor har vi bilde av noen paprikaer og chillier i forskjellige format. Kvaliteten og oppløsningen er angitt under, der alle er JPEG utenom den første som er en ukomprimert rådatafil. Nå som vi er klar over dette kan vi studere og sammenligne bildene i detalj. Vi ser en gradvis endring i bildene når JPEG elliminerer data og runder av. Bildet blir mer kornete. Om man sammenligner bildet lengst til venstre og lengst til høyre ser vi tydlige forskjeller. Bildet lengst til høyre er mye mer kornete og ser ikke like naturlig ut. Når det er sagt har vi ikke noe problem med å se konturen og helheten i bildet.

Kilde: https://www.math.cuhk.edu.hk/~lmlui/dct.pdf dato: 04.10.17

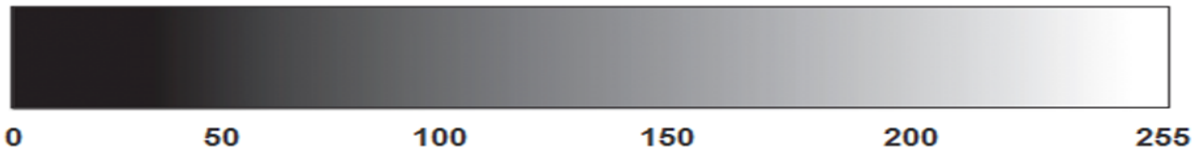
Generelt: http://www.techradar.com/news/computing/all-you-need-to-know-about-jpeg-compression-586268/2

Generelt: https://www.youtube.com/watch?v=Q2aEzeMDHMA&t=568s
https://en.wikipedia.org/wiki/Discrete_cosine_transform

Vår omgjøring fra bilde til fourierrekke

Forklaring av teoridelen

Tidligere definerte vi en fourierrekke som summen av harmoniske funksjoner med ulike frekvenser og amplituder, og er tilnærmet lik en periodisk funksjon. Nå skal vi presentere et bildet ved hjelp av en fourierrekke. For ved å bryte ned et bilde i flere pixler vil hver individuelle pixel få en egen fargeverdi. I dette eksemplet skal vi bruke fargeverdier fra og med 0 til og med 255, hvor 0 er sort og 255 er hvit. Fargeverdien definerer altså hvilken fargetone en pixel har, illustrert i figuren under.



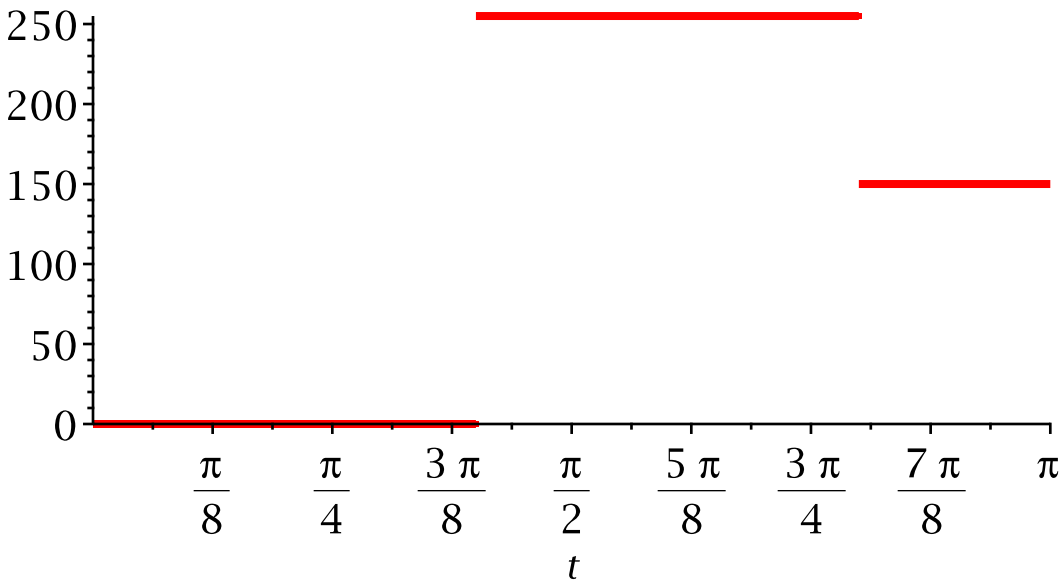
Figur 1

Nå som vi kan representere fargetonene som fargeverdier, skal vi fremstille fargeverdiene grafisk. Langs den horisontale aksen får hver pixel et like stort intervall hver, hvor alle pixlene i bildet totalt utgjøre perioden *T*. Og langs den vertikale aksen har vi fargeverdiene, som er representert gjennom amplituden til funksjonen. Når alle verdiene fra pixlene er satt inn i graf *f(t)* vil funksjonsuttrykket i det gitte intervallet være lik fargeverdien, altså en horisontal rett linje. Som et eksempel tar vi et bilde som inneholder følgene fargeverdier i kronologisk rekkefølge: 0, 255, 150, 150, 0 (sort, hvit, grå, grå, sort),. Under er *f(t)* presentert med perioden $T = \pi$.

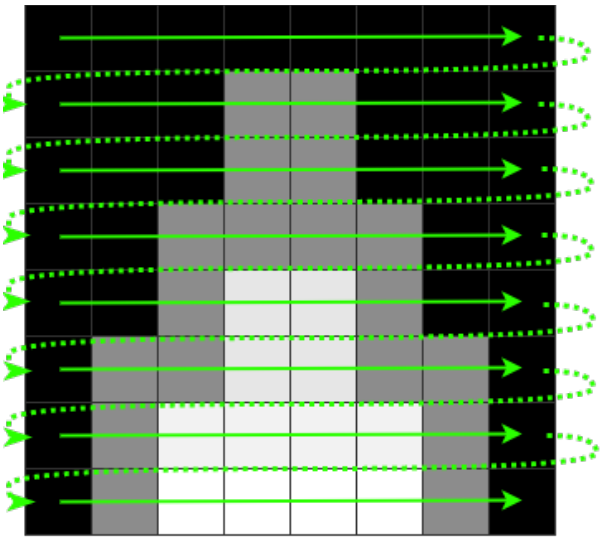
$$f := t \to piecewise\left(0 \le t \le \frac{2\,\mathrm{Pi}}{5}, 0, \frac{2\,\mathrm{Pi}}{5} \le t \le \frac{4\,\mathrm{Pi}}{5}, 255, \frac{4\,\mathrm{Pi}}{5} \le t \le \frac{6\,\mathrm{Pi}}{5}, 150, \frac{6\,\mathrm{Pi}}{5} \le t \le \frac{8\,\mathrm{Pi}}{5}, 150, \frac{8\,\mathrm{Pi}}{5} \le t \le \mathrm{pi}, 0\right);$$

$$f := t \mapsto \begin{cases} 0 & 0 \le t \le \frac{2\pi}{5} \\ 255 & \frac{2\pi}{5} \le t \le \frac{4\pi}{5} \\ 150 & \frac{4\pi}{5} \le t \le \frac{6\pi}{5} \\ 150 & \frac{6\pi}{5} \le t \le \frac{8\pi}{5} \\ 0 & \frac{8\pi}{5} \le t \le \pi \end{cases} \qquad \textbf{(4.2.1.1)}$$

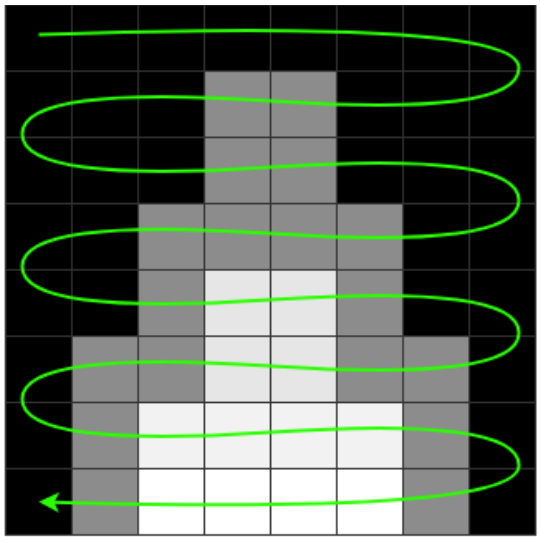
$plot(f(t), t = 0 .. \mathrm{Pi}, discont = true, symbol = \text{"point"}, color = \text{"red"}, thickness = 3)$



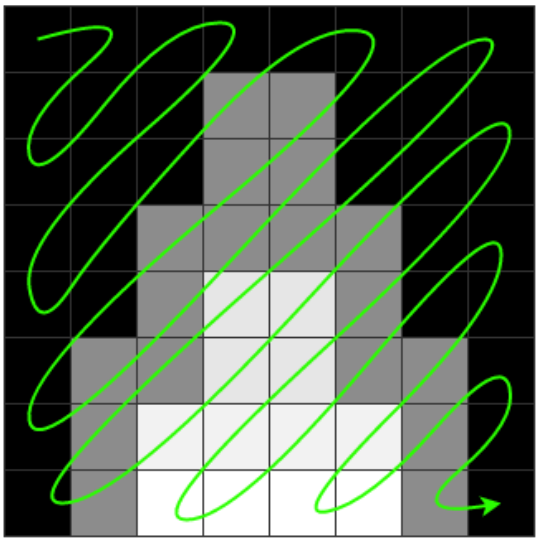
Nå som vi har sett hvordan vi kan plotte pixlene inn i grafen, må vi vite hvordan vi systematisk skal legge inn pixlene fra bildet. Vi skal finne den måten man legger inn data på hvor kurven av fourierrekken blir tilnærmet lik bildet. For eksempel, vil en løsning være å legge alle pixlene med lik fargeverdi i samlede grupper. På den måten får kurven færrest utslag, og sannsynligheten for at verdiene som er tilnærmet like bilde, blir lik de faktiske verdiene til bildet. Vi har definert tre ulike metoder som vi vil se nærmere på. Metode nr. 1 er å starte øverts i venstre hjørne og lese fra venstre mot høyre for hver linje. Metode nr. 2 starter også i øverste venstre hjørne, men for hver nye linje går man motsatt vei av forrige linje. Metode nr. 3 starter i øverste venstre hjørne og går i et sikksakk mønster diagonalt ned mot det nederste høyre hjørne. (Se Figur 2)



Metode 1



Metode 2



Metode 3

Figur 2

Anvendelse av teorien

Fremgangsmåte

Nå som vi har 3 ulike metoder for å gjøre et bilde om til en fourierrekke, så skal vi generere en fourierrekke med hver av metodene og analysere hvilken som gir den beste tilnærmingen. Under er det en liste i kronologisk rekkefølge som forteller hvordan vi skal gjøre et bilde om til en fourierrekke.

1. Lage to ulike typer bilder, et med skarpe kanter mellom lyst og mørkt og et med naturlige, myke overganger.
2. Dele bildet opp i mindre biter på 8x8 piksler, for å ikke få for mye data per fourierrekke.
3. Generere en funksjon $f(t)$ som minner om (4.2.1) som benytter seg av de 3 ulike

metodene.

4. Lage en graf som viser $f(t)$ og $\psi(t)$ med ulike antall ledd.
5. Generere et bilde utifra $\psi(t)$ og sammelikne kvaliteten med orginalen
6. Analysere resultatene for å finne ut:

- Hvilken metode som er best for ulike bildetyper
- Hvor nært originalen bildet blir
- Hvor mange ledd vi kan fjerne uten å miste for mye oppfattet kvalitet

Metode 1

Metode 2

Metode 3

Analyse

Konklusjon

5. Konklusjon

END

6. Resultat

END

7. Referanser

END