TMA4123 - Matlab
Oppgavesett 2

18.02.2013

1 Fast Fourier Transform

En matematisk observasjon er at data er tall, og ofte opptrer med en implisitt rekkefølge, enten i rom eller tid. Da er det naturlig å beskrive denne dataen som en vektor $\mathbf{y} \in \mathbb{C}^N$, hvor N er veldig stor. (Grunnen til at vi bruker $\mathbb{C}^N$ i stedet for det mer naturlige $\mathbb{R}^N$ er nettopp den raske Fourier-transformen.) [1] Fra lineæralgebraen vet vi at vektorer i $\mathbb{C}^N$ kan beskrives relativt til en basis.

Definisjon 1. *En* basis $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$ *for* $\mathbb{C}^N$ *er et sett av* N *lineært uavhengige vektorer i* $\mathbb{C}^N$*.*

Når vi har en basis, kan vi skrive en vektor $\mathbf{y}$ som en lineærkombinasjon av basisvektorene, dvs.

$$\mathbf{y} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_N\mathbf{v}_N \tag{1}$$

og dette kan bare gjøres på én måte. Vektoren som består av koeffisientene $[c_1\ c_2\ \ldots\ c_N]^\top$ kalles vektoren $\mathbf{y}$ relativt til basisen $\mathcal{B}$, og vi skriver $[\mathbf{y}]_\mathcal{B}$. Hvis vi setter vektorene i $\mathcal{B}$ sammen til en matrise $B = [\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_N]$, kan vi skrive ligning (1) på matriseform

$$B[\mathbf{y}]_\mathcal{B} = \mathbf{y}$$

En basis som er nyttig, i tillegg til standardbasisen, er *Fourier-basisen*

$$\mathcal{F} = \{\mathbf{w}_0, \mathbf{w}_2, \ldots, \mathbf{w}_{N-1}\}$$

hvor

$$\mathbf{w}_n^k = \frac{1}{N}e^{inx_k}, \quad k = 0, 1, \ldots, N-1,$$
$$x_k = \frac{2\pi k}{N}.$$

(med $\mathbf{w}_n^k$ mener vi element k i vektoren $\mathbf{w}_n$, merk at vektorene er indeksert fra 0 til $N-1$.) Faktoren $\frac{1}{N}$ er en normaliseringsfaktor, denne er ikke så viktig. For Fourier-basisen skriver vi gjerne $\hat{\mathbf{y}}$ for $[\mathbf{y}]_\mathcal{F}$.

[1] Det finnes lignende transformer, som den diskrete cosinus-transformen, som kan brukes på $\mathbf{y} \in \mathbb{R}^N$, men disse er vanligvis implementert ved hjelp av FFT.

Denne basisen er nyttig fordi vektoren

$$\mathbf{w}_n = \frac{1}{N}[1 \ e^{inx_1} \ e^{inx_2} \ \dots \ e^{inx_{N-1}}]^\top$$

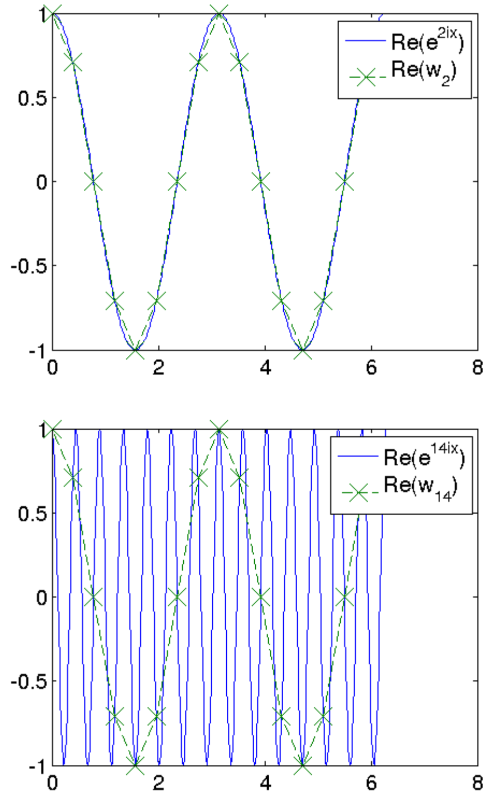
er en svingning med frekvens $\frac{n}{T}$. Hvor T er den "fysiske" lengden til $\mathbf{y}$, for eksempel varigheten av et lydsignal.

Så når vi skriver $\mathbf{y} = \sum_{n=0}^{N-1} \hat{y}_n \mathbf{w}_n$, "splitter" vi $\mathbf{y}$ i komponenter med ulik frekvens. Formelen for $\hat{\mathbf{y}}$ er

$$\hat{y}_n = \sum_{k=0}^{N-1} y_k e^{-inx_k}.$$

Algoritmen som datamaskinen bruker til å beregne $\hat{\mathbf{y}}$ kalles FFT. Antall operasjoner datamaskinen bruker på å beregne $\hat{\mathbf{y}}$ er omtrent $CN \log_2(N)$, hvor C er en konstant som avhenger av detaljer i implementasjonen. At FFT går som $N \log_2(N)$ (og ikke N^2) betyr at man kan beregne FFT av selv ganske store vektorer på brøkdelen av et sekund, selv på en vanlig laptop.

1.1 Aliasing og symmetri



Figur 1: Plot av to ulike Fourierbasisvektorer

Som vi ser i figuren vil raske svingninger “se ut som” langsomme svingninger siden vi bare sampler i et endelig antall punkter. Vi kan også vise dette fra definisjonen.

$$\mathbf{w}_{N-n}^k = \frac{1}{N} e^{i(N-n)x_k} = \frac{1}{N} e^{iNx_k} e^{-inx_k} = \frac{1}{N} e^{-inx_k} = (\mathbf{w}_n^k)^*$$

(Med z^* mener vi den komplekskonjugerte til z). Vi kunne også ha skrevet $\mathbf{w}_{N-n} = \mathbf{w}_{-n}$. Noen ganger snakker man om derfor om “de negative frekvensene”.

Et beslektet fenomen er symmetri av den Fouriertransformerte. Hvis $\mathbf{y}$ er reell, gjelder

$$\hat{y}_{N-n} = \sum_{k=0}^{N-1} y_k e^{-i(N-n)x_k} = \sum_{k=0}^{N-1} y_k e^{inx_k} = \hat{y}_n^*$$

Det motsatte holder også, hvis $\hat{y}_{N-n} = \hat{y}_n^*$, så er $\mathbf{y}$ reell. Når vi manipulerer signaler i frekvensrommet, må vi passe på at denne symmetrien holder på det vi inverstransformerer, ellers ender vi opp med ikke-reelle resultater.

2 FFT av lydsignaler i Matlab

Kommandoen `[y Fs] = wavread('foo.wav')` importerer lydfilen `foo.wav` som en vektor i Matlab. (Eller to dersom `foo.wav` er i stereo.) `wavread` returnerer også samplerate (`Fs`.) Denne trenger vi når vi skal finne ut hvilke frekvenser de ulike elementene i den Fourier-transformerte tilsvarer.

I Matlab støter vi på et problem med indekseringen. Vanligvis bruker man 0-indeksering i signalbehandling, siden formelene for Fourier-transformen og den inverse er enklere i denne indekseringen. Matlab er, som nesten det eneste programmeringsspråket, 1-indeksert. Dette må vi ta hensyn til.

Det følgende scriptet, som også ligger på hjemmesiden som `wavexample.m`, importerer filen `chord.wav`, tar beregner FFT av denne som `z` og plotter absoluttverdien av den transformerte mot frekvensene. Deretter kopierer den første halvdel av `z`, og gjenskaper symmetrien. Tilslutt tar den den inverstransformerte av `znew`. `ynew` vil være en kopi av `y`.

```
[y, Fs] =wavread('chord.wav');
z = fft(y);
N=numel(y); % antall elementer i y.
dF=Fs/N;
freq=dF*(0:N-1);
plot(freq, abs(z)); % plotter |z| mot frekvensene.

znew=zeros(N,1);
znew(1:ceil(N/2)) = z(1:ceil(N/2)); %Kopierer 1. halvdel av z.
%Erstattes med annen kode i oppgavene
znew(N:-1:floor(N/2)+2) = conj(znew(2:ceil(N/2)));
% sikrer at znew er symmetrisk.

ynew = ifft(znew);
```

Når vi skal manipulere et signal i frekvensdomenet, jobber vi hovedsaklig med den første halvdelen av den Fourier-transformerte, det vil si element 1 til $\lceil \frac{N}{2} \rceil$, og rekonstruerer den øvre delen av spekteret ved hjelp av symmetrirelasjonen som i Matlab-indeksering er $\hat{y}_{N-n+1} = \hat{y}^*_{n+1}$.

Oppgave 1 Se på plottet av Fourier-transformerte av 'chord.wav'. Hva er de dominerende frekvensene? De åpne strengene på en gitar har frekvensene Lav E - 82,4 Hz, A - 110,0 Hz, D - 146,8 Hz, G - 196,0 Hz, H - 246,9 Hz, høy E - 329,6 Hz. Ser du noe på noen av disse frekvensene?

2.1 Low- og high-pass filtere

Et low-pass filter er en algoritme som tar inn et signal, og fjerner de høye frekvensene fra signalet. Det motsatte er et high-pass filter, som fjerner lave frekvenser. Slike filtere brukes for eksempel i lydsystemer som har egne høyttalere for bass og diskant. Hvilke frekvenser man tar vare på vil variere. I den enkleste formen har man en såkalt cutoff-frekvens F_c, og tar vare på frekvenser som er lavere/høyere enn denne. Vår implementasjon av et low-/high-pass filter er som følger

1. Beregn $\hat{\mathbf{y}} = \text{fft}(\mathbf{y})$.

2. Lag vektoren $\hat{\mathbf{y}}_{\text{ny}}$ som er en kopi av $\hat{\mathbf{y}}$ for de lave/høye frekvensene, og 0 for de høye.

3. Symmetriser $\hat{\mathbf{y}}_{\text{ny}}$.

4. Beregn $\mathbf{y}_{\text{ny}} = \text{ifft}(\hat{\mathbf{y}}_{\text{ny}})$.

Oppgave 2

(a) Lag et Matlab-script som laster inn filen `elvis.wav` som utfører et low-pass filter med cutoff-frekvens 800Hz på signalet i `elvis.wav`. Du bør skrive scriptet slik at du lett kan endre inputfil og cutoff frekvens. (Dvs., sett `Fc=800`, og bruk `Fc` videre i scriptet.)

(b) Gjør tilsvarende med et high-pass filter.

Ett low-pass filter kan også brukes til støyfiltrering. Dette er basert på en observasjon av at mens "det interessante" i et signal stort sett ligger i de lave frekvensene, mens støy består av alle frekvenser. Støy som inneholder alle frekvenser i like stor grad, kalles "hvit støy".

Oppgave 3

(a) Lag et signal med tilfeldig støy ved å bruke `randn(N,1)`. Beregn og plott den Fourier-transformerte av den tilfeldige støyen. Sammenlign plottet av den Fourier-transformerte av støyen med tilsvarende plot for `chord.wav`, `chamberlain.wav`

En enkel, men nyttig, matematisk modell er at et signal med støy er summen av det opprinnelige signalet og hvit støy.

$$\mathbf{y} = \mathbf{y}_{\text{signal}} + \mathbf{y}_{\text{støy}}.$$

Om vi bare kjenner $\mathbf{y}$, er det matematisk umulig å rekonstruere $\mathbf{y}_{\text{signal}}$, men med ett low-pass filter kan vi bli kvitt deler av støyen.

(b) Modifiser Low-pass filteret du implenterte i oppgave 2 på til å bruke `chambernoise.wav` som input med cutoff frekvens på 2500 Hz.