Et lite tilleggsnotat når det gjelder (veldig) store matriser som man likevel kan regne med

Øyvind Ryan

Matematisk Institutt, UIO

September 28, 2021

Det finnes også mange andre nyttige matriser A som

- er for store til å lagre,
- har få ikke-nuller (så, langt fra en lenkematrise),
- ikke har lav rang (så, langt fra matrisen K),

men der $A\mathbf{x}$ likevel lar seg regne ut effektivt.

Ide: Finn en "algoritme" f slik at $A\mathbf{x} = f(x)$, der f regner ut $A\mathbf{x}$, på en litt annen måte (som sparer mange operasjoner) enn vanlig matrisemultiplikasjon, Algoritmen f kan for eksempel utnytte at A kan skrives som et produkt av "enkle matriser".

Diskret Fourier Transform

- I signalbehandling (se Sek. 4.7) kalles $n \times n$-matrisen F med elementer $F_{kl} = e^{-2\pi ikl/n}$ for *Fouriermatrisen*.
- $F\mathbf{x}$ kalles for Diskret Fourier Transform (DFT) av $\mathbf{x}$.
- $F\mathbf{x}$ gir en frekvensrepresentasjon av $\mathbf{x}$. Viktig for analyse av feks. lyd.
- I Matlab kan man regne ut $F\mathbf{x}$ ved hjelp av kommandoen `fft`, som er mye mer effektivt enn å regne ut $F\mathbf{x}$ direkte.

Koden på neste side sammenligner tidsbruk for $F\mathbf{x}$ og `fft(x)`.

Matlabkode (sml_fft.m)

```
n=10000;
x=rand(n,1);
k=0:(n-1); l=0:(n-1);
F = exp( - 2*pi*i*k'*l/n );

tic;
fft(x);
toc;

tic;
F*x;
toc;
```

Pythonkode (sml_fft.py)

```
import random
from numpy import *
import time

n=10000
#x=[1,1,1,1,1]
x=random.random(n)
k=matrix(arange(n))
l=matrix(arange(n))
F = exp( - 2*pi*1j*k.T*l/n )

start=time.time()
fft.fft(x)
end=time.time()
print(end-start)

start=time.time()
F*reshape(x,(n,1))
end=time.time()
print(end-start)
```

Hvorfor er `fft(x)` mye bedre når n vokser?

Det viser seg at matrisen F kan faktoriseres

$$F = F_1 \cdots F_k$$

Der hver F_i har mange nuller! `fft(x)` regner ut $F\mathbf{x}$ ved å bruke denne faktoriseringen i stedet.