Rask introduksjon til programmering (i MatLab)

Arne Stormo

1 Hva er programmering

En datamaskin er en komplisert kalkulator som kan gjøre kompliserte beregninger og lagre data elektronisk. Dette gjøres ved å flippe elektroner i minnet og harddisken i retninger som maskinen tolker som *av* eller *på*, eller mer kjent som 0 eller 1. Posisjoner og kombinasjoner av disse 0 og 1 verdiene tolkes av maskinen og leses ut som nyttig informasjon for oss dødelige mennesker.

Alle operasjoner gjøres slavisk av datamaskinen slik den får beskjed om. Å gi datamaskinen et sett med instruksjoner kalles å *programmere* maskinen. Det finnes flere måter gi en maskin instruksjoner på. Den mest fundamentale måten er å direkte be maskinen flippe hver enkelt elektron ved bruk av maskinkode. Dette er en langdryg og komplisert affære, og er ikke en spesiell intuitiv teknikk. Derfor er det utviklet flere typer *høynivå* programmeringsspråk. Dette er språk som er lettere for mennesker å forstå, og dermed er det lettere å be maskinen gjøre nettopp det du ønsker den skal gjøre. Når instruksjonene er skrevet, oversettes dem til maskinkode som datamaskinen forstår. Dette er en prosess som kalles *kompilering*.

MatLab er et program som oversetter programmeringskode til maskinkode. Først skrives instruksjoner på det språket MatLab er laget til å forstå. Dette skrives inn i et tekstdokument. Deretter mates disse instruksjonene inn i MatLab, som oversetter linje for linje hva som er skrevet i dokumentet om til maskinkode, som deretter blir utført.

NB: Datamaskiner er stokk dumme, og gjør utelukkende det du ber dem om. Derfor må man være sikker på at det man faktisk ber den om å gjøre, var det man mente å be den om å gjøre.

2 Konsepter i programmering

I bunn og grunn er det svært få operasjoner en datamaskin gjør. Den kan lagre verdier i minnet, utføre elementære matematiske operasjoner ($+$, $-$, $\times$ og $\div$) og foreta *logiske* operasjoner, dvs. sjekke om et uttrykk er sant eller ikke (f.eks $2 < 5$ er sant, mens $2 > 5$ er usant). Det er når mange av disse enkle operasjonene kombineres at store ting kan utrettes.

For å skrive et enkelt program er det viktig å ha fått med seg enkelte vesentlige elementære konsepter.

2.1 Grunnleggende operasjoner

En datamaskin er som sagt intet annet en en komplisert kalkulator. Den foretar enormt mange enkle beregninger i en forrykende fart. I MatLab kan vi be maskinen gjøre disse beregningene svært enkelt. Linjen

```
3+2
```

gir utskriften

```
ans = 5
```

på skjermen. Plusstegnet blir av MatLab oppfattet som en *operator*. Dermed ser MatLab hvilken verdi som står før og etter operatoren, og gjør den operasjonen som tilhører nettopp denne operatoren. Utregningen $1 + 2 \times 5 - 3 \div 2$ vil i MatLab bli skrevet som

```
1+2*5-3/2
```

og gi utskriften

```
ans = 9.5000
```

til skjermen.

2.2 Variabler

Hvis minnet til datamaskinen kan sammenlignes med et arkiv fullt av skuffer, så kan en variabel sammenlignes med en av skuffene. Hver skuff har en merkelapp, og kan inneholde en enkel verdi. En skuff kan med andre ord ha merkelappen 'antall epler i posen' og inneholde heltallet '7'. Hvis noen senere vil vite nettopp hvor mange epler det ligger i posen, så kan de sjekke i skuffen med den merkelappen.

Det er mulig å lage en skuff med navnet 'var1' uten at den inneholder noen informasjon. For å bruke denne variabelen til noe fornuftig må vi *tilordne* en verdi til variabelen. Det vil si at man legger et stykke informasjon ned i skuffen med merkelappen 'var1'.

I MatLab tilordner man en verdi til variabelen i det man erklærer den. Hvis du for eksempel ville lagt flyttallet, dvs. desimaltallet (maskiner tolker heltall og flyttall som to forskjellige ting), '4.2' i variabelen 'var1' måtte man ha skrevet

```
var1=4.2
```

Dette vil resultere i at 4.2 er lagret i 'var1' og at

```
var1=4.2000
```

blir skrevet ut på skjermen. (Hint: skriv et semikolon ; på slutten av linjen, og verdien blir kun lagret, ikke skrevet til skjerm.)

Deretter kan man på ny tilordne hvilken som helst verdi til 'var1'. I matlab trenger man ikke erklære hvilken type variabel 'var1' er. Dette forandrer seg automatisk etterhvert som man tilordner nye verdier. I andre, mer kompliserte språk, må man erklære hvorvidt en variabel kan inneholde heltall, flyttall, tekst (kalt strenger i programmeringsverden) eller logiske uttrykk (true/false dvs. sant/usant).

Variabler, kombinert med de elementære operasjonene danner grunnstenen i all programmering.

Et enkelt eksempel: vi har to poser med epler 'pose1' og 'pose2'. Vi vil vite hvor mange epler vi har totalt og lagre det i 'epler' dette gjøres slik

```
pose1=7;
pose2=5;
epler=pose1+pose2;
```

Variabelen 'epler' innehar nå verdien 12. Vi kan tilordne nye verdier av en variabel, basert på hva den forrige var. Si at vi legger til to epler i den første posen. Denne tilordningen vil se slik ut

```
pose1=pose1+2;
```

'pose1' vil nå inneha verdien 9.

Til sist er det verdt å nevne at på samme måte som en skuff kan få en merkelapp, så kan hele reoler av skuffer få en felles merkelapp. Dette kalles lister, *arrays* eller andre navn avhengig av hvilket språk man bruker. For å illustrere hvordan dette virker, så kan vi kalle en reol av minnet for 'fotballspillere' og lagre alderen til de forskjellige fotballspillerne i hver sin skuff. Dette vil gjøres slik:

```
fotballspillere(1)=25;
fotballspillere(2)=23;
fotballspillere(3)=27;
osv...
```

For å hente opp alderen til keeperen vil man skrive

```
fotballspillere(1)
```

som vil resultere i at

```
ans = 25
```

blir skrevet til skjermen.

2.3 Funksjoner

For relativt kompliserte beregninger som f.eks. $\sin\theta$ må maskinen bruke de enkle operatorene $+$, $-$, $\times$ og $\div$. Dette fordi maskinen er bygd opp slik at den kun kan håndtere slike enkle operasjoner. Da må $\sin\theta$ beregnes ved hjelp av en rekkeutvikling.

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \dots$$

Dette er slitsomt å skrive hver gang en skal beregne en sinus. Heldigvis så er denne kodesnutten allerede skrevet av de som lagde MatLab. Den ligger lagret i et *bibliotek* lastet opp av MatLab. Den spesifikke kodesnutten som beregner sinus kalles en *funksjon*. Å benytte seg av en slik kodesnutt heter å *kalle en funksjon*. Det gjøres på følgende måte.

```
a=3.151593/2;
sin(a)
```

eller bare

```
sin(3.151593/2)
```

som gir utskriften

```
ans = 1.00000
```

Her er funksjonen man kaller sin() og *argumentet* er variabelen a, eller bare tallet 1.5707965. En funksjon kan ha flere argumenter. Man kan lagre verdien funksjonen gir eller *returnerer*. for å lagre verdien av $\sin\frac{\pi}{2}$ i en variabel skriver man

```
var=sin(1.5707965);
```

Funksjoner kan gjøre andre ting enn å gjøre matematiske uttrykk. F.eks sørger funksjonen disp(*arg*) for at argumentet blir skrevet til skjerm. Et annet eksempel er funksjonen plot(*arg1*,*arg2*), som *plotter* argument 2 mot argument 1 i et koordinatsystem. Nøyaktig hvordan forskjellige funksjoner fungerer står beskrevet i MatLabs dokumentasjon på nettet. Eventuelt kan man skrive 'help *funksjonsnavn*' i kommandovinduet for en beskrivelse av funksjonen.

2.4 Logiske setninger

Logiske setninger er linjer i kode som sjekker hvorvidt et uttrykk er sant eller ikke, og dermed velger hvordan programmet skal håndtere løpet videre avhengig av svaret. I matlab blir '0' tolket som usant eller *false* og alle andre tall som sant eller *true*. Dvs. $3<1$ får verdien 0, mens $3>1$ blir 1. Når MatLab kjører gjennom kommandoene en har skrevet ned og kommer til en logisk setning, sjekker den hvorvidt argumentet er sant eller ikke, og hopper til den linjen i koden som er anvist avhengig av resultatet av sjekken. Funksjonen som sjekker dette heter *IF(arg)*. En IF setning må avsluttes med kommandoen 'end' Et enkelt eksempel på bruk av dette er

```
a=1;
b=1;
if(a)
  b=b+1;
end
disp(b);
```

Dette vil skrive 2 til skjermen. Hadde 'a' vært 0, ville 1 ha blitt skrevet til skjermen. Hvis man skal velge mellom to forskjellige handlinger avhengig av kondisjonen sant/usant vil man behøve kommandoen *ELSE*. Et eksempel blir da

```
a=1;
b=1;
if(a)
  b=b+1;
else
 b=b-1;
disp(b);
```

Her vil fortsatt 2 bli skrevet til skjerm, men hadde 'a' vært 0, så ville 'b' blitt 0, og 0 hadde blitt skrevet til skjerm.

2.5 Løkker

En *løkke* eller *loop* er en sekvens i en kode som skal gjentas med små eller ingen forandringer fra gang til gang. Det finnes flere typer løkker, men i hovedsak brukes det to forskjellige typer; *FOR* og *WHILE* løkker. FOR løkker har en forhånds satt sett med *iterasjoner* eller repetisjoner av løkken. Det gjøres ved å lage en liste som inneholder like mange elementer som antall ganger du skal iterere. En løkke som skriver ut alle tallene fra en til ti lages ved å skrive

```
for i=1:10
  disp(i);
end
```

I MatLab lager kommandoen 1:10 en liste med ti elementer med verdene en til ti. Legg merke til at løkker også må avsluttes med 'end'. En WHILE løkke har en logisk sjekk i argumentet En WHILE løkke som gjør det samme som FOR løkken over vil se slik ut:

```
i=1
while(i<11)
  disp(i);
  i=i+1;
end
```

Det finnes mange nyttige ting man kan utrette ved hjelp av løkker. Det er også mulig å ha løkker og logiske sjekker inni hverandre. Dette kallen nøsting, og da gjelder det å ha tunge rett i munnen på hvor en setter 'end' kommandoene.

3 Eksempler

3.1 Hello World!

Verdens enkleste og kanskje det første programmet brukt når man tar i programmering er *Hello World*. Dette er en liten kodesnutt som får maskinen til å skrike 'Hello World!' til deg som operatør når du kjører det.

Koden:

```
disp('Hello World!');
```

Tekstbiten eller strengen er erklært inne mellom de to apostrofene.

3.2 Fibonaccirekken

Fibonaccirekken er en rekke tall som starter med to ett tall, og som fortsetter som summen av de to forrige tallene. Vi kan skrive ut de 20 første tallene i fribonaccirekken med et enkelt program. Denne inneholder en logisk setning nøstet i en FOR løkke.

Koden:

```
var1=1;
var2=1;
var3=1;
for i=1:20
  if(i<3)
    disp(1);
  else
    var3=var2;
    var2=var2+var1;
    var1=var3;
    disp(var2);
  end
end
```

3.3 Grafen x^2

For å tegne eller *plotte* grafen x^2 fra -4 til 4 vi må først lage to lister. En med x-verdiene og en med x^2-verdiene. Deretter må vi kalle funksjonen plot() slik at grafen blir tegnet.

Koden:

```
x=-4:0.1:4
index=1
for i=x
  y(index)=x(index)^2;
  index=index+1;
end
plot(x,y);
```

Dette vil resultere i en graf der x^2 for $x = [-4, 4]$.