En innføring i MATLAB for STK1100

Matematisk institutt
Universitetet i Oslo
Februar 2017

1 Innledning

Formålet med dette notatet er å gi en introduksjon til bruk av MATLAB. Notatet er først og fremst beregnet for studenter som tar STK1100, men vil også kunne være nyttig for andre. Notatet ble opprinnelig utarbeidet av Peter Acklam og er senere revidert av Ørnulf Borgan.

2 Variabler, skalarer, vektorer og matriser

En variabel er en "lagringsenhet" som kan inneholde tall eller noe annet. En variabel med navn `a` som inneholder en skalar (ett enkelt tall), kan opprettes med

```
>> a = 3.14
```

Her er `>>` promptet som angir hvor brukeren skal skrive inn teksten (selve promptet skal ikke skrives). Videre er `=` tilordningsoperatoren, altså variabelen `a` tilordnes (eller gis) verdien 3.14. Når dette er skrevet inn, og `[Enter]`-tasten trykket, så vil MATLAB svare med å skrive

```
a =
    3.1400
```

Dersom det hadde stått et semikolon (`;`) på slutten av linja som ble skrevet inn, så ville resultatet ikke blitt skrevet ut på skjermen.

En kan bruke piltastene for å navigere fram og tilbake på kommandolinjen og for å bla fram og tilbake i tidligere kommandoer.

Det er mulig å putte flere tall i én variabel ved å bruke en vektor. En vektor er ganske enkelt en liste av tall. En radvektor kan for eksempel lages ved å skrive

```
>> b = [1, 2, 3, 4, 6, 4, 3, 4, 5]
```

MATLAB svarer da med

```
b =
     1     2     3     4     6     4     3     4     5
```

Kommaene er valgfrie, men for lesbarhetens skyld kan det være greit å ha dem med.

En kan hente ut ett eller flere elementer av en vektor ved å bruke en eller flere indekser

```
>> b(5)

ans =
     6

>> b([5, 7])

ans =
     6     3
```

Det femte elementet i `b` er 6 og det syvende elementet er 3. Merk at siden resultatet ikke lagres i noen variabel, så blir resultatet lagret i en standardvariabel som kalles `ans` (forkortelse for "answer").

En søylevektor kan for eksempel lages ved å skrive

```
>> c = [1; 2; 3]
```

MATLAB svarer da med

```
c =
     1
     2
     3
```

Merk at kommaer (eller mellomrom) brukes for å skille elementer horisontalt, mens semikolon brukes for å skille elementer vertikalt. Disse kan kombineres til å lage matriser

```
>> d = [1, 2, 3; 4, 6, 4]
```

som gir

```
d =
     1     2     3
     4     6     4
```

En kan få bruk for å lage vektorer og matriser hvor alle elementene har samme verdi, for eksempel bare nuller eller enere. Funksjonene som brukes til det er `zeros`, `ones`, og generelt `repmat`. For eksempel

```
>> f = zeros(2, 3)
```

gir

```
f =
     0     0     0
     0     0     0
```

og

```
>> g = repmat(3.14, [2, 3]);
```

gir

```
g =
    3.1400    3.1400    3.1400
    3.1400    3.1400    3.1400
```

Det første argumentet til `repmat`, `3.14`, er verdien som skal gjentas, og det andre argumentet, `[2, 3]`, er størrelsen på den ønskede matrisen – i dette tilfellet 2×3.

Noe en også kan ha bruk for er å lage vektorer med tall som øker i like store trinn, som for eksempel $1, 2, 3, \ldots$ og $0, 0.1, 0.2, \ldots$ Til dette kan en bruke kolon-operatoren. For å lage en vektor fra `min` til `max` hvor hvert element øker med 1, brukes `min:max`, for eksempel

```
>> 3:6

ans =
     3     4     5     6
```

For at hvert element skal øke med verdien `step`, brukes `min:step:max`, for eksempel

```
>> 3:0.5:6

ans =
    3.0000    3.5000    4.0000    4.5000    5.0000    5.5000    6.0000
```

3 Aritmetiske operatorer

Operatorene `+` og `-` virker i MATLAB som en er vant til fra matematikken, men for multiplikasjon, divisjon og potens skiller MATLAB mellom operasjoner som skal gjøres elementvis og matriseoperasjoner

`.*`	elementvis multiplikasjon	`*`	matrise-multiplikasjon
`./`	elementvis divisjon	`/`	matrise-divisjon
`.^`	elementvis eksponent	`^`	matrise-eksponent

Vi vil her konsentrere oss om de elementvise operasjonene. Merk at vi får dem ved å bruke operatorene med . foran, altså de i venstre kolonne ovenfor. At en operasjon utføres elementvis vil si at samme operasjon utføres på hvert enkelt element i en vektor.

Her er et eksempel på elementvis addisjon

```
>> b = 2 + [1, 2, 3]

b =
     3     4     5
```

Tallet 2 legges til hvert element i vektoren.

Her er et eksempel på elementvis multiplikasjon

```
>> b = [1, 2, 3] .* [4, 5, 6]

b =
     4    10    18
```

Legg merke til hvordan hvert element i den ene vektoren multipliseres med tilsvarende element i den andre vektoren.

Her er et eksempel på å ta eksponenten elementvis

```
>> b = [1, 2, 3] .^ 2

b =
     1     4     9
```

Skriv `help arith` og `help slash` for flere detaljer om aritmetiske operatorer.

4 Elementære funksjoner

MATLAB har alle de vanlige elementære funksjonene `sin`, `cos`, `exp`, osv. De virker elementvis, slik at for eksempel `sin` anvendt på en vektor, vil være det samme som å anvende `sin` på hvert enkelt element. Her er et par eksempler

```
>> exp(1)

ans =
    2.7183

>> cos([0, pi])

ans =

     1    -1
```

Andre vanlige funksjoner er `sqrt` (kvadratrot), `log` (naturlig logaritme) og `log10` (logaritme med grunntall ti).

I tillegg finnes for eksempel funksjonene `abs` (absoluttverdi), `sign` (fortegnsfunksjonen), `round` (avrunding til nærmeste heltall), `fix` (avrunding mot null), `floor` (avrunding mot $-\infty$) og `ceil` (avrunding mot $+\infty$).

Skriv `help elfun` for flere funksjoner og detaljer, eller se hjelpesiden for de aktuelle funksjonene, for eksempel `help sqrt` for å se hjelpesiden til funksjonen `sqrt` (kvadratrotfunksjonen).

5 Logiske operatorer

MATLAB har følgende elementvise operatorer for sammenligning av tall

`<`	mindre enn	`>`	større enn
`<=`	mindre enn eller lik	`>=`	større enn eller lik
`==`	er lik	`~=`	er ikke lik

Disse returnerer 0 eller 1 avhengig av om resultatet er sant eller usant. For eksempel

```
>> 3 < 2

ans =
     0

>> 5 >= 4

ans =
     1
```

MATLAB har også følgende logiske operatorer

`&` og `|` eller `~` ikke

For å se om tallet i variabelen `x` ligger i det halvåpne intervallet [0,1) kan en bruke

```
>> (0 <= x) & (x < 1)
```

Skriv `help relop` for flere detaljer.

6 Andre funksjoner for vektorer

MATLAB har en del nyttige funksjoner for behandling av vektorer. Her er noen av dem

`sum`	sum	`min`	minimum
`prod`	produkt	`max`	maximum
`cumsum`	kumulativ sum	`sort`	sortering
`cumprod`	kumulativt produkt	`find`	finn indekser for sanne verdier
`length`	antall elementer	`diff`	differens av påfølgende verdier

Se `help datafun` for flere funksjoner og detaljer.

7 Enkel generering av stokastiske variabler

MATLAB har et stort antall funksjoner for å generere tilfeldige tall. Skriv `help stats` for å se en liste av alle statistikk-relaterte funksjoner. Funksjonene som genererer tilfeldige tall står under overskriften Random Number Generators.

For eksempel, for å simulere terningkast kan en bruke `unidrnd` som trekker fra en diskret uniform fordeling. Ti terningkast kan simuleres med

```
>> x = unidrnd(6, [1, 10])

x =
     3     6     2     3     6     4     6     1     3     4
```

Det første argumentet (`6`) angir øvre grense for tallene som det skal trekkes fra, i dette tilfellet $\{1, 2, \ldots, 6\}$, og det andre argumentet `[1,10]` angir at vi ønsker 10 elementer (eller mer presist: `1` rad og `10` kolonner).

Vi kan finne de kastene som ga en sekser ved å bruke for eksempel

```
>> x == 6
```

```
ans =

     0     1     0     0     1     0     1     0     0     0
```

Her er det et ett-tall for alle kast som ga en sekser og en null alle andre steder. Vi kan finne indeksene til kastene som ga sekser ved å bruke

```
>> find(x == 6)

ans =

     2     5     7
```

og fra dette kan vi finne hvor mange kast som ble utført fra vi fikk en sekser til neste gang vi fikk en sekser, ved å bruke

```
>> diff(find(x == 6))

ans =

     3     2
```

Disse verdiene kan også sees direkte fra sekvensen av nuller og enere over. Det er to nuller (altså to kast som ikke ga sekser) mellom de to første ett-tallene, og det er en null (altså ett kast som ikke ga sekser) mellom de to siste ett-tallene.

8 Deskriptiv statistikk

MATLAB har en rekke funksjoner for deskriptiv statistikk. Her er noen av dem

`sum`	sum	`min`	minimum
`mean`	gjennomsnitt	`max`	maximum
`var`	varians	`median`	median
`std`	standardavvik	`iqr`	kvartildifferanse
`cov`	kovarians	`range`	variasjonsbredde
`corr`	korrelasjon	`prctile`	persentil (kvantil)

Disse kommandoene genererer 1000 standard normalfordelte variabler og beregner deres gjennomsnitt og standardavvik

```
>> x = normrnd(0, 1, [1, 1000]);
>> m = mean(x)

m =
    0.0455

>> s = std(x)

s =
    1.0313
```

9 Plotting

MATLAB har et stort utvalg av funksjoner for plotting. De mest vanlige er

`plot`	plotte linjer og kurver
`scatter`	plotte punkter
`hist`	histogram
`bar`	søylediagram
`mesh`	3-dimensjonalt gitterplott
`surf`	3-dimensjonalt flateplott

Her er et eksempel som plotter tettheten for en standard normalfordeling som en rød heltrukken linje

```
>> x = -4:0.1:4;                      % lag vektor med x-verdier
>> y = normpdf(x);                    % lag vektor med y-verdier
>> plot(x, y, 'r-');                  % plott med roed linje
>> xlabel('x'); ylabel('tetthet');    % gi navn til aksene
>> title('Standard normalfordling');  % gi figuren en tittel
>> box on                             % sett ramme rundt figuren
>> grid on                            % legg til stiplete linjer
```

Kommentar-tegnet i MATLAB er `%` og alt som stå etter det hopper MATLAB over.

Og her er et eksempel som plotter sannsynligheten for å få x rette i lotto dersom en spiller 8 rekker. Dette kan sammenlignes med at vi har en eske med 34 kuler hvorav 7 er sorte og resten hvite. Hvis vi trekker ut 8 kuler, hvor stor er sannsynligheten for at x av disse er sorte?

```
>> m = 34;                    % antall kuler i alt
>> n = 7;                     % antall sorte kuler
>> k = 8;                     % antall som trekkes ut
>> x = 0:7;                   % antall sorte blant de som trekkes ut
>> y = hygepdf(x, m, k, n);   % beregn sannsynligheter
>> bar(x, y);                 % lage stolpediagram
>> xlabel('k'); ylabel('sannsynlighet');
>> title('Sannsynlighet for x rette');
```

For å kontrollere plottingen finnes andre nyttige funksjoner

`axis`	juster aksene	`title`	lag en tittel
`hold`	overskriv plott, ikke erstatt	`xlabel`	sett navn på x-aksen
`grid`	legg linjer i aksesystemet	`ylabel`	sett navn på y-aksen
`box`	lag en boks rundt aksesystemet		

Se hjelpeteksten for de respektive funksjonene for mer informasjon om hvordan de brukes, for eksempel `help plot`.