En kort innføring i MATLAB

Matematisk institutt
Univeristetet i Oslo
Januar 2004

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 er utarbeidet av Peter Acklam (2001) og senere revidert av Ørnulf Borgan (2004).

2 Starte og avslutte MATLAB

MATLAB kjøres under Linux (Unix) på PC-ene på Mat.nat.fakultetets terminalstuer. Se `www.matnat.uio.no/it/student/termstuer.html`

2.1 Starte MATLAB i Unix

Åpne et nytt terminalvindu. Dette kan gjøres ved å skrive

```
xterm &
```

I det nye terminalvinduet startes MATLAB ved å skrive

```
matlab
```

Brukergrensesnittet som startes har tre vinduer. Det er i vinduet til høyre at kommandoer skal skrives inn. Dette brukergrensesnittet er dessverre ganske tungt og kan være noe tregt. Det kan derfor være bedre å kjøre MATLAB direkte i terminalvinduet ved å starte MATLAB med

```
matlab -nojvm
```

2.2 Avslutte MATLAB

MATLAB avsluttes med `quit` eller `exit`.

3 Variabler, skalarer, enkle vektorer og matriser

3.1 Konstruksjon av variabler, skalarer, vektorer og matriser

En variabel er en lagringsenhet som kan inneholde tall og 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. Prøv dette selv.

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 slags 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 er det 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 lages ved å skrive for eksempel

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

MATLAB svarer da med

```
c =
     1
     2
     3
```

Altså: Kommaer 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
```

Skriv `help punct`, `help horzcat` eller `help vertcat` for flere detaljer.

3.2 Opplisting og sletting av variabler

For å liste opp alle variabler som er laget, bruk funksjonen `who` eller `whos`:

```
>> who

Your variables are:

a         b         c         d

>> whos
  Name      Size         Bytes  Class

  a         1x1              8  double array
  b         1x9             72  double array
  ...       ...            ...  ...

Grand total is 19 elements using 152 bytes
```

For å slette en eller flere variabler brukes `clear`.

Skriv `help who`, `help whos` eller `help clear` for flere detaljer.

3.3 Mer om konstruksjon av vektorer og matriser

Noe en ofte får bruk for er 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.

Skriv `help zeros`, `help ones`, `help repmat` for flere detaljer.

Noe en også ofte har 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 brukes kolon-operatoren eller funksjonen `linspace`. 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
```

Funksjonen `lispace` gjør det samme, men i stedet for å gi skrittlengden, gis antall elementer som ønskes, for eksempel

```
>> min = 2; max = 4; n = 5;
>> linspace(min, max, n)

ans =

    2.0000    2.5000    3.0000    3.5000    4.0000
```

Legg merke til at i eksemplet over er flere uttrykk satt på samme linje. Det er ingenting i veien for å gjøre det, men merk at dersom det er semikolon etter et uttrykk, så vil resultatet ikke bli skrevet ut, mens dersom det er komma vil resultatet bli skrevet ut.

4 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. Vi vil konsentrere oss om de elementvise operasjonene. Dersom du ikke vet hva matriseoperasjoner er for noe (eller matriser, for den del), så ikke tenk mer på det – bare husk å bruke operatorene med . foran, altså de i venstre kolonne.

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

At en operasjon utføres elementvis vil si at samme operasjon utføres på hvert enkelt element i en vektor.

```
>> b = 3.14 + 2

b =
    5.1400
```

Her er et eksempel på elementvis addisjon. Tallet 2 legges til hvert element i vektoren.

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

b =
     3     4     5
```

Her er et eksempel på elementvis multiplikasjon. Legg merke til hvordan hvert element i den ene vektoren multipliseres med tilsvarende element i den andre vektoren

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

b =
     4    10    18
```

Her er et eksempel på å ta eksponenten elementvis.

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

b =
     1     4     9
```

Addisjon og subtraksjon utføres *alltid* elementvis.

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

5 Elementære funksjoner

MATLAB har alle de vanlige elementære funksjonene `sin`, `cos`, `exp` osv. Disse 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 $+\infty$) osv.

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).

6 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.

7 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.

8 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. 10 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.

9 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`	kvartildifferans
`cov`	kovarians	`range`	variansbredde
`corr`	korrelasjon	`prctile`	persentil (kvantil)

Se i utskriften for `help stats` under overskriften Descriptive Statistics for flere detailjer. Navnene på funksjonene for statistiske fordelinger består alle av et prefiks som angir fordelingen (`unid` for diskret uniform fordeling, `norm` for normalfordeling, `hyge` for hypergeometrisk fordeling, osv.) og et suffiks som angir hva en ønsker fra fordelingen (`rnd` for tilfeldige tall, `pdf` for sannsynilghetstetthet, `cdf` for kumulativ fordeling, `inv` for invers kumulativ fordeling, osv.).

Dette 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
```

10 Plotting

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

`plot`	plotte linjer, kurver og punkter
`hist`	histogram (også `histc`)
`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 r{\o}d linje
>> xlabel('x'); ylabel('tetthet');    % sett merkelapper p{\aa} akser
>> title('Standard normalfordling');  % sett tittel p{\aa} figuren
>> box on                             % sett ramme rundt figuren
>> grid on                            % legg til stiplete linjer
```

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

```
>> m = 34;                    % antall baller i alt
>> n = 7;                     % antall r{\o}de baller
>> k = 8;                     % antall som trekkes ut
>> x = 0 : 7;                 % antall r{\o}de blant de som ble trukket ut
>> y = hygepdf(x, m, k, n);   % beregn sannsynligheter
>> bar(x, y);                 % lage s{\o}ylediagram
>> xlabel('k'); ylabel('sannsynlighet');
>> title('Sannsynlighet for x rette');
```

Kommentar-tegnet i MATLAB er `%`.

Andre funksjoner relatert til figurer er

`figure`	åpne et figurvindu
`clf`	lukk gjeldende figurvindu
`close`	lukk angitt figurvindu
`print`	skriv ut eller lagre figuren

For å kontrollere plottingen finnes andre nyttige funksjoner

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

For å sette tekst i en figur brukes for eksempel `text` og `gtext`.

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

11 Lesing og skriving av data til fil

Den mest brukte funksjonen for å lese data fra fil er `load`. Den kan lese vanlige tekstfiler med tall og filer i MATLABs eget format (filer som slutter på `.mat`). Hvis vi har en fil som heter `data.txt` og som inneholder for eksempel

```
40.6  93.5  91.7  41.0  89.4
 5.8  35.3  81.3   1.0  13.9
20.3  19.9  60.4  27.2  19.9
```

så kan denne leses inn i MATLAB og lagres i variabelen `X`, ved å bruke

```
>> X = load('data.txt');
```

For å skrive data til fil kan en bruke `save`. Funksjonen `save` lagrer vanligvis data i MATLABs eget format, men en kan lagre i tekstformat ved å kalle `save` med `-ascii` til slutt. For eksempel, dersom en vil lagre dataene i variabelen `X` til tekstfilen `data.txt`, så kan en bruke

```
>> save data.txt X -ascii
```

Se `help save` for flere detaljer.

12 Programfiler

Det kan ofte være greit å putte kommandoer i en fil dersom de skal kjøres om og om igjen med bare små endringer mellom hver gang. Vi kan for eksempel lage en fil som heter `prog.m` som inneholder

```
n = 100;                          % antall tilfeldige tall
x = normrnd(0, 1, [1, n]);        % generer tilfeldige tall
hist(x)                           % plott histogram
```

Denne filen må ligge slik at MATLAB finner den. Det enkleste er å legge filen på det samme filområdet som MATLAB ble startet fra. I Unix vil MATLAB alltid finne filene som ligger på filområdet `~/matlab/`, så det kan være et greit sted å legge slike filer.

For å utføre funksjonene i filen `prog.m` skriver en

```
>> prog
```

og så vil MATLAB utføre innholdet av `prog.m`. For å se innholdet av en fil i MATLAB bruk funksjonen `type`, for eksempel `type prog.m`.

En kan også legge filer på andre filområder, men da må filområdene legges til MATLABs søkesti. Dette kan for eksempel gjøres ved å bruke funksjonen `addpath` (se avsnittet nedenfor om oppstartfil).

13 Diverse

13.1 Hvordan tall vises på skjermen

MATLAB bruker internt alltid full (dobbel) presisjon, men det er ikke alltid at alle sifrene vises på skjermen, og det er ikke alltid at standardformatet er hensiktsmessig. Hvordan resultatene vises kan kontrolleres med funksjonen `format`. Her er noen eksempler:

```
>> x = [0 2.71828182845905 3.14159265358979e123];
>> format short f; x

x =
  1.0e+123 *
         0    0.0000    3.1416

>> format short g; x

x =
             0       2.7183  3.1416e+123
```

Skriv `help format` for flere detaljer.

13.2 Oppstart- og avslutningsfiler

Det hender en ønsker å gjøre personlige tilpasninger, for eksempel bruke et annet format for visning av tall enn det som er standard (ved å bruke `format` som vist over) eller gjøre flere `m`-filer tilgjengelige for MATLAB ved å utvide MATLABs søkeområde (gjøres med `addpath`). Det kan da være greit å gjøre dette automatisk ved å legge funksjoner og kommandoer inn i en oppstartfil.

Den personlige oppstartfilen utføres når MATLAB starter og heter `startup.m`. I Unix skal denne filen ligge på filområdet `~/matlab/`.

Tilsvarende finnes en avslutningsfil som blir utført når MATLAB avsluttes. Den heter `finish.m` og skal ligge samme sted som oppstartfilen.

13.3 Manøvrering

En kan enkelt manøvrere i kommandovinduet ved å bruke piltastene opp og ned og fram og tilbake (eller kontroll-taster for dem som fortrekker det).

Ved å skrive litt av en kommandolinje og så pil opp, så vil MATLAB lete gjennom alle tidligere kommandolinjer og skrive ut den første som begynner med den angitte teksten. Dersom pil opp brukes flere ganger, vil MATLAB vise alle foregående kommandoer som begynner med den angitte teksten, en etter en.

Ved å bruke tab-tasten, vil MATLAB fylle ut så mye den klarer av det en allerede har skrevet inn. Ved å trykke tab-tasten en gang til, vil MATLAB vise en liste med alternativer.

13.4 Vise en skjermfull av gangen

Dersom utskriften i kommandovinduet farer over skjermen, kan man få MATLAB til å vise en skjermfull av gangen ved å bruke `more on`. Ved å bruke `more off` vil denne funksjonaliteten skrus av igjen.

14 Hjelp til selvhjelp

MATLAB har et par online-demonstrasjoner som kan prøves

`intro`	slideshow som viser enkel bruk av MATLAB
`demo`	velg mellom flere demonstrasjoner

MATLAB har også et omfattende hjelpesystem

`lookfor`	søk gjennom file etter nøkkelord (for eksempel `lookfor random`)
`help`	online-hjelp for funksjoner (for eksempel `help sort`)
`helpwin`	online-hjelp i eget vindu
`doc`	HTML-hjelp (for eksempel `doc sort`)
`helpdesk`	omfattende hjelpesystem i HTML-format

14.1 Informasjon på web-en

Det finnes en god del informasjon om MATLAB på web-en. Her er et par adresser

`http://www.mit.edu/~pwb/cssm/`	The MATLAB FAQ
`http://www.mathworks.com/`	The MathWorks, Inc.
`http://www.mathworks.com/search/`	søkeside
`http://www.mathworks.com/support/`	support