Introduksjon til Matlab

Håvard Berland

INSTITUTT FOR MATEMATISKE FAG — NTNU 2003

Innhold

1 Introduksjon

Dette kurset gir en liten gjennomgang av hva en student trenger for å komme i gang med MATLAB ved Institutt for matematiske fag, NTNU.

Målgruppe for kurset er studenter som nettopp har begynt i 3. klasse Industriell matematikk. Bakgrunnskunnskaper er da programmering i Java, med innslag av numerikk.

Materialet er det meningen man skal kunne gå gjennom på 1-2 timer, og man kan godt lese det helt på egen hånd, samtidig som man har MATLAB startet på en datamaskin foran seg.

Kurset inneholder mange kodeeksempler. Disse skrives med "Courier"-skrifttypen som ser `slik` ut. Dersom en linje begynner med et dollartegn `$` så betyr det at det er en kommando du skal skrive inn i en xterm-vindu (terminalvindu, kommandolinjevindu) hvor du ikke har startet andre programmer (egentlig betyr det at du kan skrive bash/skall-kommandoer). Hvis linja starter med `>>` så er det en MATLAB-kommando som kommer etterpå. Kommentarer inni koden starter med prosenttegn `%` som er MATLABs kommentartegn. Du kan gjerne skrive dette som kommandoer i MATLAB, men MATLAB vil ignorere dem.

2 Oppstart av MATLAB

Vi har MATLAB i versjon 6.5 tilgjengelig på alle våre unix-maskiner, både datamaskinene på salene og på serverene. Til å begynne med anbefaler vi å starte MATLAB på maskinen du sitter ved, og ikke på serverene. Du gjør dette ved å enten skrive `matlab` i et lokalt xterm-vindu som du har åpnet enten via høyreklikk-menyen, eller knapperaden nederst til høyre.

```
$ matlab
```

Det er også mulig å starte MATLAB fra høyreklikk-menyen. Vi har med vilje ikke installert MATLAB på Windows-serveren vår som er tilgjengelig fra alle datamaskiner.

MATLAB har et grafisk grensesnitt (GUI) som du vil få om du ikke sier i fra om noe annet. Dette er ikke nødvendig, og du kan klare deg like fint uten, da det går mye raskere å bruke MATLAB uten dette. For å starte matlab uten GUI inne i xterm'en du har åpnet, skriver du

```
$ matlab -nojvm
```

Etterhvert som du skal gjøre større og større beregninger, må du etterhvert kjøre MATLAB på serverene eller må forlate datamaskinen du sitter på og la den regne ferdig på egenhånd. Da må du starte matlab på en snill måte i forhold til andre programmer som måtte kjøre på datamaskinen. Du gjør dette ved å skrive `nice` framst på kommandlinja

```
$ nice matlab -nojvm
```

men dette er ikke nødvendig mens du går gjennom dette kurset.

3 Skalarer, vektorer og matriser

MATLAB er en forkortelse for *Mat*rix *lab*oratory , og er derfor sentrert rundt matriser. Det er dette den kan best. Skalarer, vektorer og matriser er egentlig samme ting, av MATLAB

kalt *arrays* (tabell) (noen kaller slikt tensorer), bare av forskjellig dimensjon. Skalarer er 0-dimensjonale, vektorer er 1-dimensjonale, matriser er 2-dimensjonale når man snakker om *arrays* (ikke bland dette med "dimensjonen til en matrise eller en vektor", som er antall elementer i matrisa/vektoren). Vårt vanlige vokabular stopper her, men det gjør ikke MATLAB.

Tilordning av skalar:

```
>> x = 4.2
x =
    4.2000
```

Hvis du ikke vil se utskriften av en kommando du skriver, så legger du til et semikolon bakerst på linja. Dette er mest aktuelt når du skriver script som det kommer mer om senere.

```
>> x = 4.2;
>>
```

Tilordning av en liggende vektor;

```
>> b = [ 1 2 3 4]
b =
     1     2     3     4
```

Du kan også skille disse elementene inni klammeparantesen med et komma om du ønsker det. Hvis du heller vil ha en stående vektor, har du to alternativer. Enten kan du transponere en liggende vektor, eller du kan taste den inn stående.

```
% første alternativ, transponere en liggende vektor med operatoren '
>> b = [ 1 2 3 4]'
b =
     1
     2
     3
     4
% eller skrive den inn stående. Rader skilles med semikolon.
>> b = [ 1 ; 2 ; 3 ; 4]
b =
     1
     2
     3
     4
```

Tilordning av en 4 ganger 4-matrise gjøres nå analogt med vektorene, da en matrise kan sees på som en stående vektor av liggende vektorer.

```
>> A = [ 1 2 3 4 ; 5 6 7 8 ; 9 10 11 12 ; 13 14 15 16 ]
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

Tips: Når du selv skriver inn linjene ovenfor, så vil du få flere tomme linjer enn det som er med her. Det er fordi vi her har skrevet kommandoen `format compact`. Dette får MATLAB til å kutte ut bruken av tomme linjer.

4 Grunnleggende operasjoner

4.1 Addering

Addering av skalarer, vektorer eller matriser skriver du på den mest åpenbare måten, ved hjelp av tegnet +. Det du legger sammen *må* ha samme dimensjon, hvis ikke vil du få en feilmelding.

For vektorer og matriser fungerer addisjon elementvis

```
>> 1 + 3
ans =
      4
>> b'
ans =
     1     2     3     4
>> b' + b'
ans =
     2    4    6   8
```

For subtraksjon bytter du ut pluss med minus. Den fungerer akkurat likt.

4.2 Multiplikasjon

Multiplikasjon har litt mer substans for MATLAB enn addisjon. Multiplikasjon fungerer ikke elementvis i MATLAB uten at du sier i fra. For skalarer virker multiplikasjon slik du vil forvente den, men for vektorer og matriser vil den hele tiden tenke matrisemultiplikasjon.

Så for å gange sammen to matriser eller en en matrise med en vektor er det bare å taste rett inn

```
% vår tidligere matrise A opphøyd i andre potens
>> A * A
ans =
    90   100   110   120
   202   228   254   280
   314   356   398   440
   426   484   542   600
% Matrisa A ganget med den stående vektoren b, og omgjort til en liggende
% vektor for å spare skjermplass
>> (A*b)'
ans =
    30    70   110   150

% Når b er en stående vektor, er b' en liggende vektor:
>> b'
ans =
     1     2     3     4
% Vi kan ikke gange sammen A med en liggende vektor:
>> A*b'
??? Error using ==> *
Inner matrix dimensions must agree.
```

Hvis du vil gange sammen to vektorer eller matriser element for element, kan du bruke operatoren `.*`, dette heter elementvis multiplikasjon. Omvendt heter det også elementvis

divisjon som har operatoren `./`. Bare divisjonssymbolet anvendt på vektorer og/eller matriser vil gi deg en feilmelding, da man matematisk sett ikke kan dele på vektorer eller matriser. Hvis du derimot forsøker å dele en vektor på en skalar med divisjonstegnet, så vil MATLAB forstå at du mener elementvis divisjon, det samme gjelder for multiplikasjon.

4.3 Ligningsløsing

For å løse et ligningssystem med koeffisienter i matrisa A og høyreside i vektoren b, løser vi for x i ligningen $Ax = b$. Du vet fra Matematikk 3 at denne løsningen kan vi finne som $x = A^{-1}b$. Når man gjør slikt på datamaskiner, er det som regel en uting å regne ut inverser til matriser. Man gjør heller Gauss-eliminering som du også lærte om i Matematikk 3, eller bruker enda mer sofistikerte iterative metoder som er enda raskere. I MATLAB bruker du heller "backslash"-operatoren $\backslash$ enn den inverse til ei matrise for å løse slike ligningssystemer, men vi kan også bruke den inverse direkte som vi skal se under.

Matrisa A som vi har brukt tidligere, gir ingen unik løsning til problemet $Ax = b$. Det er fordi den er singulær, har bare rang 2 (må ha 4 for dette problemet), eller har determinant lik null.

```
>> A
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
>> rank(A)
ans =
     2
>> det(A)
ans =
     0
>> x = A\b
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.387779e-18.
(Type "warning off MATLAB:nearlySingularMatrix" to suppress this warning.)
x =
         0
         0
         0
    0.2500
```

Advarselen om at du ikke kan stole på resultatet skal du stole på. Dette resultatet er bare tull, siden matrisa A *er* singulær , som vi har vist både ved å finne rangen , og ved å se på determinanten .

Vi tester da heller med en tilfeldig 4×4-matrise

```
>> B = rand(4)
B =
    0.4103    0.8132    0.1987    0.0153
    0.8936    0.0099    0.6038    0.7468
    0.0579    0.1389    0.2722    0.4451
    0.3529    0.2028    0.1988    0.9318
```

```
% sjekker at matrisa B ikke er singulær:
>> det(B)
ans =
   -0.1344
% OK, da kan vi løse Bx = b:
>> x = B\b
x =
   -3.7773
    2.2553
    3.2532
    4.5383
```

4.4 Vektor- og matriseindeksering

Vektorindeksering av matriser er en nyttig og kraftig funksjonalitet ved MATLAB, som det anbefales å skjønne og lære seg. Dette fordi det kan gi enorme besparelser i form av brukt cpu-tid.

Hvis du vil ha tak i element nummer tre i en vektor c, skriver du

```
>> c = [ 3 1 5 9 ]'
ans =
     3   1   5   9
>> c(3)
ans =
     5
```

altså med paranteser rundt indeksen du vil ha tak i. Ikke bland sammen hakeparanteser og vanlige paranteser her. Men MATLAB takler også å få en vektor inn som argument innenfor parantesene. Da tolker den det som en vektor av indekser. La oss si du vil ha ut indeks 2 og 3 i vektoren c som har fire elementer;

```
>> indekser = [ 2 3 ]
indekser =
     2     3
>> c(indekser)
ans =
     1
     5
```

Det samme gjelder og blir enda mer nyttig for matriser, her må du spesifisere et indekssett/en indeksvektor for hver dimensjon (2 dimensjoner for matriser). Hvis du vil ha ut en hel kolonne, rad, kan du bruke et kolon `:` for å symbolisere hele kolonnen, raden. For å raskt lage indeksvektorer, bruker du også kolon, med eller uten et hoppargument i midten, som her:

```
>> indekser = 1:10
indekser =
     1     2     3     4     5     6     7     8     9    10
>> annenhver = 1:2:10
annenhver =
     1     3     5     7     9
```

Så for å indeksere i vår matrise A fra tidligere, så kan vi gjøre for eksempel disse operasjonene:

```
>> A(2,2)
ans =
     6
>> A(:,2)
ans =
     2
     6
    10
    14
>> A(2,:)
ans =
     5     6     7     8
>> A(2:3,2:3)
ans =
     6     7
    10    11
```

Samme indeksering kan vi også bruke for å tilordne deler av matriser til like verdier (eller til vektorer/matriser, om du får til det!). La oss si du trenger en 10×10 matrise med 1 langs alle kantene og bare 0 inni. Dette kan du da gjøre for eksempel slik:

```
>> C(1:10,1:10) = 1;
>> C(2:9,2:9) = 0
C =
     1     1     1     1     1     1     1     1     1     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     1     1     1     1     1     1     1     1     1
```

Vi kan legge inn vektoren b inni her i kolonne 3 om vi er nøye med indeksene:

```
>> C(4:7,3) = b
C =
     1     1     1     1     1     1     1     1     1     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     1     0     0     0     0     0     0     1
     1     0     2     0     0     0     0     0     0     1
     1     0     3     0     0     0     0     0     0     1
     1     0     4     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     0     0     0     0     0     0     0     0     1
     1     1     1     1     1     1     1     1     1     1
```

5 Programmering

5.1 Matlab-filer, funksjoner og script

Etterhvert som du begynner å skrive MATLAB-kode, er det kjekt å samle sekvenser av kommandoen er i egne filer. MATLAB-filer slutter på `.m`, og blant disse filene, skiller MATLAB mellom to typer, *script* og *funksjoner* .

Et script er ei fil som kun inneholder en sekvens av MATLAB-kommandoer som om du skulle tastet dem direkte inn i matlab (men i fila har du selvsagt ikke med tegnene » som kommandoeksemplene her har med). Hvis du vil kjøre innholdet av et script i MATLAB, sørger du for at du står i samme katalog som du har lagret scriptet (med filendelse `.m`) og taster inn filnavnet *uten endelsen*. Gitt du har scriptet `world.m` som gjør diverse beregninger, og som til slutt skriver ut et tall, kan du kalle det slik:

```
>> world
ans =
     42
```

Funksjoner lagres også i filer som slutter på `.m`, men starter med en linje `function y = f(x)` hvor `y` er en variabel du tilordner i funksjonen som blir gitt som returverdi, og `x` er argumenter til funksjonen din. Du kan også ha en kommaseparert liste over input-variabler. Mer informasjon finner du med kommandoen `help function`.

Et eksempel på en funksjon som beregner $y = x^2$:

```
function y = f(x)
y = x.^2;
```

Dette lagres i fila `f.m`, og du kan så kalle funksjonen med for eksempel `f(4)` i MATLAB.

Her har vi brukt potens-operatoren `^`. Legg merke til at vi satte et punktum foran `^`. Dette betyr akkurat som for multiplikasjon, elementvis forhøying til gitte potens. Følgen av det, er at funksjonen vår til og med tillater at input-argumentet `x` er en vektor, og den vil så returnere en ny vektor `y` der alle elementene er tilsvarende element i `x` kvadrert. Alternativet her hadde vært å programmert en `for`-løkke, men måten vi gjorde det på, kan gå noe sånt som 10 ganger raskere. Semikolon etter kommandolinjer også lurt å ha med i koden, ellers vil du få mye uinteressant utskrift av verdier hver gang du bruker funksjonen.

5.2 Editor

For å skrive MATLAB-funksjoner og -script, så trenger du en editor. Du kan bruke hvilken som helst editor for å lage disse filene, men *emacs* anbefales. Du kan starte emacs på ei fil fra unix-kommandolinja med

```
$ emacs f.m
```

og så skrive hva du vil inni der. Emacs vet om at filer som slutter på `.m` er MATLAB-filer og gir deg fargelegging og også muligheten for å kjøre script i MATLAB direkte fra emacs.

MATLAB har også sin egen editor, som du kan starte fra MATLAB-kommandolinja med kommandoen `edit f.m`. Denne er ikke like god som emacs, siden Mathworks (de som lager MATLAB) er best på å lage matematisk programvare, ikke teksteditorer, men det kan hende du synes denne likevel er bedre integrert med resten av MATLAB.

5.3 `for`-løkker og `if`-setninger

Her viser vi bare hvordan `for`-løkker og `if`-setninger fungerer. MATLAB har de fleste vanlige andre kontrollstrukturer du er vant med fra Java.

Prinsippet for `for`-løkker er at de løper over en vektor. Slike kontroll-strukturer som `for`-løkker gjør du vanligvis i enten script eller i funksjoner, det er litt klønete å skrive dem inn på kommandolinja (men det går!). Syntaksen er `for variabel=vektor; gjør noe; end`. Det du taster inn for `variabel` vil bli tilorndet hvert element i `vektor` i tur og orden. For eksempel

```
for a=1:10
  b(a) = a+a;
end
```

vil resultere i en vektor `b` med element nummer i satt til $2i$. Hvis du fikk med deg det som tidligere er nevnt om vektorindeksering og elementvise operasjoner, forstår du at vi heller bør lage denne vektoren med `b = 2*(1:10)`. Hver gang du lager en `for`-løkke, bør du tenke gjennom om du kan unngå denne ved bruk av vektorindeksering og elementvise operasjoner.

`if`-setninger utfører et sett instruksjoner hvis en betingelse er tilfredstilt. Syntaksen er `if test; noen instruksjoner; end`. En test kan enten skrives ved sammenlignings-operatorer, f.eks. `==` (som betyr at to element er like, viktig forskjell på `==` og `=`, tilordning), `<`, `>`, `<=`, `>=`, eller `~=` (ulik). Du kan også teste på om en skalar er ulik 0, da er testen sann. Et eksempel er

```
if det(A)~=0
   inv(A)
elseif
   disp("Matrisen er singulær")
end
```

6 Enkel plotting

MATLABs `plot`-kommando kan egentlig ikke annet enn å plotte to *vektorer* mot hverandre. Du kan ikke gi en funksjon inn til MATLAB, kun hva funksjonen er på noen vektorverdier. Så lenge funksjonene du bruker kan ta vektorer inn som argument (slik eksempelfunksjonen `f` ovenfor kunne), er dette likevel ikke noe problem. La oss si vi vil plotte funksjonen $y = x^2$, representert ved funksjonen `f(x)` for x i intervallet 0 til 2. Vi må selv spesifisere hvor "fint" plottet skal være, det vil si for mange datapunkter vi skal ta med. MATLAB lager en linje mellom hvert datapunkt.

```
>> x = 0:0.01:2;
>> plot(x,f(x))
% antall datapunkter vi har brukt, kan vi finne med size:
>> size(x)
ans =
     1   201
% som betyr at vektoren x har 1 rad og 201 kolonner
```

Etter at du har laget et plott, og vil ha det inn i en annen applikasjon, må du eksportere plottet til et filformat som andre program kan bruke. Encapsulated Postscript (eps) er som regel det beste formatet for slikt. For å eksportere figuren din til fila `figur.eps`, kan du mens plotvinduet er åpent skrive

```
>> print -depsc2 'figur.eps'
```

Hvis du vil lagre figurene dine i pdf-format for bruk sammen med pdflatex, ikke la Matlab eksportere til pdf. Eksporter til eps som over og bruk `epstopdf` fra kommandolinja for å konvertere til pdf. Matlab lager hele A4-sider med figuren din på, og det lar seg vanskelig lime pent inn i andre dokument.

7 Flere kommandoer

I stedet for å gå gjennom flere kommandoer i dette kurset, lister vi heller opp navn på relevante kommandoer listet opp tematisk. Les MATLAB sin egen hjelp om hver av disse kommandoene med `help kommandonavn`.

7.1 Vektoroperasjoner

Disse funksjonene tar som regel en vektor inn, og gir noen tall ut. Prefikset "cum" betyr kumulativ. `diff` kan mellom annet brukes til å gjøre numerisk derivasjon (men du må tenke selv for å bruke den), og spesielt `find` er ekstremt kraftig til å behandle store matriser i forbindelse med vektorindeksering.

```
sum, prod, abs, cumsum, cumprod, max, min, length, sort, diff, find,
mean, std, any, all
```

7.2 Matriser

For å enkelt lage matriser i stedet for å lage løkker som gjør det samme. Også noen funksjoner som kan ta matriser inn

```
zeros, rand, ones, eye, diag, size, norm, trace, null, orth, cond, eig,
expm, inv, lu, qr, rref, cond, rank, det, sparse, rot90
```

7.3 Variabler

Disse kan du bruke til å holde oversikt over hvilke variabler du har allokert, og slette variabler du ikke trenger. Jobber du med svære matriser, kan det av og til være fordel å slette dem fra minnet, slik at du ikke bruker det opp.

```
who, whos, clear, save, load
```

7.4 Elementære funksjoner

Disse er vanlige kommandoer du skulle kjenne igjen. `fix`, `floor` og `ceil` er relatert til avrunding.

```
exp, log, cos, sin, tan, asin, acos, atan, sqrt, abs, sign, fix, floor,
ceil, round
```

7.5 Plotting

Du vil vanligvis også bruke noen av disse funksjonene for å sette opp plottene dine pent, eller for å lage 3D-plot.

```
xlabel, ylabel, zlabel, title, legend, text, grid, loglog, semilogx,
semilogy, hold, box, mesh, surf, bar, quiver, meshgrid, contour, shading,
subplot, figure, set, axis
```

7.6 Flytkontroll

Dette er kontrollstrukturer som du bruker sammen med `for` og `if` som vi allerede har nevnt

```
switch, while, continue, break, return
```

7.7 Andre kommandoer

```
disp, error, input, tic, toc, format, print, nargin, nargout, lookfor
```

8 Mer hjelp

8.1 MATLAB innebygd hjelp

Det du kommer til å bruke mest av hjelpefunksjoner, er MATLAB-kommandoen `help`. For de aller fleste kommandoer, kan du bare skrive

```
>> help inv

 INV    Matrix inverse.
    INV(X) is the inverse of the square matrix X.
    A warning message is printed if X is badly scaled or
    nearly singular.

    See also SLASH, PINV, COND, CONDEST, LSQNONNEG, LSCOV.
```

Hvis hjelpeteksten er så lang at den scroller over hele terminalvinduet ditt, kan du skrive kommandoen `more on` først for å få MATLAB til å stoppe hver gang den når enden av vinduet.

8.2 Referansemanual

Vi har også mye referansedokumentasjon på MATLAB liggende på webserveren vår. Se på
`http://www.stud.math.ntnu.no/matlab`
for flere pekere. Du finner også referansedokumentasjon om du skriver inn `doc` i MATLAB. Denne kommandoen vil åpne en browser (opera er valgt på matematikk) med samme referansedokumentasjon.

Register