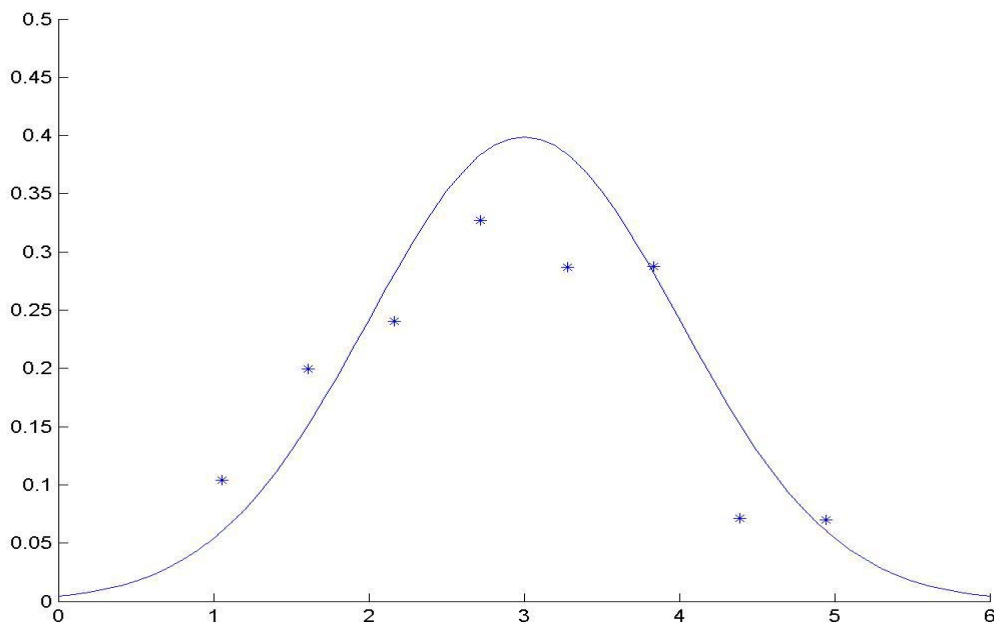


# En kort introduksjon til MATLAB

Lars Snipen og Jan Søreng

9. februar, 2004



## **Innledning.**

MATLAB er et høynivå dataprogram som er godt egnet til beregninger og visualisering. Programmet har stor utbredelse både i industri og akademia, og brukes ofte i kombinasjon med andre programmer.

Notatet gir en enkel innføring. Stikkord er grunnleggende regneoperasjoner, matriseregning, løsning av likninger, visualisering, lagre og hente data osv. Det forutsettes at leserne har noe kjennskap til matriser. I notatet er det henvisninger til en bok som betegnes Pratap. Det er følgende bok:

**Rudra Pratap**, *Getting started with MATLAB 6*, Oxford University Press, 2002.

Dette notatet er kommet i mange versjoner. Det er gjort generelle endringer, men også blitt tilpasset forskjellige kursopplegg.

## Oversikt over temaer i notatet.

1. Bli kjent med MATLAB. Vi skal gjøre oss kjent med kommando-vinduet, figurvinduet og editoren
2. Vektorer, matriser, løse likningssystemer
3. 2D-plotting
4. 3D-plotting og diverse plottemuligheter
5. Lagre og hente data.
6. Mere matrisemanipulasjon
7. Enkel programmering
8. Egne funksjoner
9. Hjelpemulighetene i MATLAB
10. Et utvalg av MATLABs muligheter for viderekommende

# 1. Bli kjent med MATLAB.

Vi skal gjøre oss kjent med kommandovinduet, figurvinduet og editoren. Bruken av vinduene vil bli belyst ved enkle eksempler.

Se Pratat, side 8

Regneoperasjoner introduseres. Det vil bli vist hvordan kommandoer og programmer utføres.

## 1.1 Kommandovinduet.

I dette vinduet skriver vi inn kommandoer. Kommandoene skrives inn etter `>>`. De blir utført ved å trykke på Enter. Semikolon etter kommandoen gjør at svaret ikke synes på skjermen. Vi viser bruk av kommandovinduet ved eksempler:

### Kommandoer:

### Kommentarer:

```
>> 2 + 3
```

```
>> 4*3
```

```
>> exp(2)
```

```
>> x = 2
```

```
>> y = x^2 + 3
```

```
>> v = [ 2 3 8]
```

v blir en radvektor.

```
>> w = [2; 3; 4]
```

w blir en kolonnevektor

```
>> w = [ 2  
        3  
        4]
```

Alternativ måte å lage kolonnevektoren w.

```
>> t = 0 : 0.1 : 1
```

t blir lik vektoren

```
[ 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 ]
```

```
>> y = sqrt(t)
```

y blir en vektor med samme lengde som t der hvert element er lik kvadratroten av det korresponderende element i t.

*Merknad:* Vi kan ofte velge om vi legger inn blanke karakterer ( dvs. tomrom ) for å øke lesbarheten. Eksempel: `2+3` og `2 + 3` gir samme resultat.

Se neste merknad for et eksempel der vi ikke kan bruke mellomrom.

*Merknad:* En kommando kan fortsettes på neste linje ved å sette ... ( tre punktum uten mellomrom ) på slutten av den linjen som skal fortsette videre.

## 1.2 Figurvindu.

Vi illustrerer bruk av figurvinduet ved et eksempel. Utfør følgende i kommandovinduet:

```
>> t = 0 : 0.1 : 6;  
>> y = sin(t);  
>> plot(t, y)
```

I første linje lages en vektor av t-verdier ( $t = [0 \ 0.1 \ 0.2 \ \dots \ 6]$ ).  
I andre linje lages en vektor som inneholder sinusverdiene til t-verdiene.  
Siste linje lager grafen av  $\sin(t)$  i figurvinduet.  
Hvis det ikke angis noe annet, så lager MATLAB grafer ved å trekke streker mellom punktene. ( Senere i notatet forklares flere muligheter )

## 1.3 Editor.

Program lages i MATLAB-editoren. Et program består av kommandoer.  
Eksempel:

```
t = 0: 0.2 : 4;  
y = 1 + exp(-t);  
plot(t, y, 'r')
```

Ved å utføre dette programmet blir alle kommandoene utført på en gang.  
Programmet plottes funksjonen  $y = 1 + \exp(-t)$  på intervallet  $[0, 4]$  og grafen blir rød.  
Her følger en beskrivelse av hvordan dette programmet lages og utføres:

Velg New på menyen File ( mer presist, velg New M-file ). Alle programfiler må slutte med etterstavelsen .m  
Slike filer kalles m-filer. MATLAB-editoren setter på etterstavelsen selv.

Se 1.6.4, side 12 i Prata for mer informasjon om filtyper

Skriv inn programmet i editoren.

Før det lagres, må det opprettes en passende mappe som filen legges i:

Hvis MATLAB ligger på maskinen som benyttes, er det vanlig at den opprettede mappen legges i mappen Work. Dette er i utgangspunktet en tom mappe ( inneholdt i mappen MATLAB ). Den er beregnet på lagring av egne programmer.

På datasalene benyttes en nettversjon av MATLAB. Da bør den opprettede mappen legges på et hjemmeområde. ( NB! Når nettversjon benyttes, så vil enkelte programmer ikke virke hvis de er lagret på et hjemmeområde. I disse situasjonene opprettes en mappe på datamaskinen som benyttes, slik at det aktuelle program kan lagres der. Husk da alltid å lagre en kopi på et hjemmeområde. )

Det er ofte lurt å opprette undermapper for de forskjellige applikasjonene. Da unngås navnekonflikt og bedrer oversikten.

Programfilen lagres i en passende mappe. Vi kan for eksempel kalle filen program1. Etterstavelsen .m legger MATLAB til automatisk. Bruk ikke -, . eller liknende i et filnavn. Begynn filnavn med en bokstav.

Deretter går vi tilbake til MATLAB-vinduet. Dette kan gjøres på forskjellige måter. Trykk for eksempel på knappen nederst på skjermen som svarer til kommandovinduet ( den har påskrift MATLAB ). Alternativt, klikk et eller annet sted i vinduet.

MATLAB-vinduet inneholder i utgangspunktet 4 vinduer: Command Window, Workspace, Current directory and Command history. Vinduene Current directory og Workspace ligger på samme sted. Man kan velge hvilket av de to som er aktivt, ved å klikke på henholdsvis Current directory eller Workspace. ( Disse vinduene kan lukkes. Hvis man ønsker å få tilbake standardoppsettet med alle fire vinduene på plass, så velges View på kommandolinjen, deretter Desktop Layout og til slutt Default. )

Aktiver vinduet Current directory. Vi må deretter finne og åpne mappen som den aktuelle programfil ligger i.

Deretter utføres programmet ved å skrive navnet på programmet i kommandovinduet og trykke på Enter. ( Navnet må skrives uten etterstavelsen .m )

Merknad: Vi kan også komme til riktig mappe i kommandovinduet ved å bruke DOS-kommandoer. Her forklares hvordan:

Se side 15 og 16 i Pratap for mer informasjon.
--

```
>> cd ..           ( change directory et hakk oppover )
>> cd mappenavn   ( change directory til den angitte mappe )
>> pwd            ( gir present working directory )
>> dir            ( gir oversikt over undermapper og filer i den
                  mappen vi er i )
```

Eksempel på bruk av kommandoene cd, pwd og dir:

```
>> pwd            ( Gir beskjed om hvilken mappe vi er i. )
>> dir            ( Gir beskjed om filer og undermapper )
```

Anta en av undermappene heter uke51

```
>> cd uke51       ( Flytter til mappen uke51 )
```

Vi ombestemmer oss og vil tilbake igjen. Dette gjøres slik:

```
>> cd ..
```

Eksempel som viser bruk av kommentarer:

```
% Programmet legger sammen tallene fra 1 til 10.
% Programmet er laget 12.12.01 av Hans Hansen.
    sum = 0;
    for i = 1:10
        sum = sum + i;
    end
    sum          % Viser innholdet av sum.
```

Kommentarer begynner med %.

Kommentarene før programkoden kan vi få ut på skjermen i kommandovinduet ved å skrive `help filnavn` (uten etterstavelen `.m`). På den måten kan vi få oversikt over hva filer inneholder. ( `help filnavn` virker kun når vi i kommandovinduet har flyttet oss til mappen der filen ligger. )

*Merknad:* En viktig kommando er `clear all`. Da fjernes alle variable ( og funksjoner ) fra MATLAB's arbeidsområde. Vi kan da begynne på nytt. Denne kommandoen er noe av det første vi benytter når vi får uforklarlige resultater eller feilmeldinger. Et vanlig problem er at samme variabelnavn brukes i flere sammenhenger. Derfor er det ofte lurt å bruke flere bokstaver i navn på variable, gjerne meningsfylte navn som forteller hva variablene representerer.

Se Pratap, side 15

*Merknad:* Ved å holde Ctrl-tasten inne og trykke på C så avbrytes MATLAB-programmer. ( Det er alltid lurt å vite hvordan man stopper )

## 2. Matriser og matriseoperasjoner.

### 2.1 Hvordan vektorer og matriser lages.

Se Pratap, side 49 – 61

```
>> u = [ 2  5  8]          ( v blir en radvektor av lengde 3 )
```

```
>> u = [ 2, 5, 8]         ( Samme resultat som foregående kommando )
```

```
>> v = [ 2;  5;  8]       ( v blir en kolonnevektor av lengde 3 )
```

```
>> v = [ 2
        5
        8 ]               ( Samme resultat som foregående kommando )
```

```
>> M = [ 2 3 4; 5 7 8]    ( M blir en 2 x 3 matrise )
```

```
>> M = [ 2 3 4
        5 7 8]           ( Samme resultat som foregående kommando )
```

Verdien til element  $j$  til en vektor  $v$  kan bestemmes ved å skrive  $v(j)$ .

Verdien til elementet  $(i, j)$  til en matrise  $M$  kan bestemmes ved å skrive  $M(i, j)$ .

Rad i til M kan bestemmes slik  $M(i, :)$  ( : representer alle verdiene til andre indeks )  
 Kolonne j til M kan bestemmes slik  $M(:, j)$  ( : representer alle verdiene til første indeks )  
 Her er noen eksempler i det vi antar v og M er definert som over.

```
>> v(2)                ( blir lik 5 )
>> M(2,3)              ( blir lik 8 )
>> M(:,3)              ( blir lik kolonne 3 i M )
>> M(1,:)              ( blir lik rad 1 i M )
```

*Merknad:* Kommandoen  $v = a:d:b$  gir vektoren  $v = (a, a + d, a + 2d, \dots, a + nd)$  der n er største hele tall slik at  $a + nd$  er mindre eller lik b. Kommandoen  $v = \text{linspace}(a, b, n)$  kan også benyttes.  
**2.2 Elementvise regneoperasjoner.**

Operatorene  $*$ ,  $/$ ,  $\backslash$  og  $\wedge$  betyr multiplikasjon, høyre divisjon, venstre divisjon og eksponering.

Se Pratap, side 58

Ved å sette et punktum foran disse operatorene ( $.*$ ,  $./$ ,  $\backslash$  og  $.\wedge$ ), så virker de elementvis på matriser ( og vektorer ). Operasjonene  $A.*B$ ,  $A./B$  og  $A.\backslash B$  er veldefinerte kun når matrisene A og B har samme størrelse. Vi belyser dette ved eksempler.

```
>> A = [ 1  2  1
         10 20 10 ];
>> B = [ 2  4  2
         20 20 20 ];
>> A.*B
```

Dette er elementvis matrisemultiplikasjon. Matrisen  $A.*B$  blir lik

$$\begin{bmatrix} 2 & 8 & 2 \\ 200 & 400 & 200 \end{bmatrix}$$

Hvert element i  $A.*B$  framkommer ved å multiplisere de korresponderende elementene i A og B.

```
>> A./B
```

Dette er elementvis ( høyre- ) divisjon. Matrisen  $A./B$  blir lik

$$\begin{bmatrix} 0.5000 & 0.5000 & 0.5000 \\ 0.5000 & 1.0000 & 0.5000 \end{bmatrix}$$

Hvert element i  $A./B$  framkommer ved å ta det korresponderende element i A og dele på det korresponderende element i B.

```
>> A.\B
```

Dette er elementvis ( venstre- ) divisjon. Matrisen  $A.\backslash B$  blir lik

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 1 & 2 \end{bmatrix}$$

Hvert element i  $A.\backslash B$  framkommer ved å ta det korresponderende element i B og dele på det korresponderende element i A.

```
>> A.^2
```

Matrisen  $A.^2$  blir lik  $\begin{bmatrix} 1 & 4 & 1 \\ 100 & 400 & 100 \end{bmatrix}$

Matrisen  $A.^2$  blir laget ved å ta hvert element i A og opphøye i annen potens.

### 2.3 Vanlige matriseoperasjoner.

Se Pratap, side 57

To matriser kan adderes og subtraheres hvis de har samme dimensjon.  $c*M$  der  $c$  er et tall og  $M$  en matrise, gir en ny matrise der hvert element i  $M$  er multiplisert med  $c$ .

$A*B$  er vanlig matrisemultiplikasjon av matrisene A og B. Her er et eksempel:

```
>> A = [ 1 2 1
        10 20 10 ];
>> B = [ 2 1
        1 1
        1 1 ];
```

```
>> A*B          ( Matrisen A*B blir lik [ 5 4
                                           50 40 ] )
```

$A^n$  er lik matrisen A ganget med seg selv n ganger. ( A må være en kvadratisk matrise. )

### 2.4 Funksjoner anvendt på matriser.

Se Pratap,  
side 61 - 63

Her er en oversikt over noen funksjoner i MATLAB:

```
mean(x)
std(x)
var(x)
cov(x)
corrcoef(x)
```

Statistiske funksjoner.

```
log(x)          ( Logaritmen med e som grunntall )
log10(x)        ( Logaritmen med 10 som grunntall )
log2(x)         ( Logaritmen med 2 som grunntall )
exp(x)
sqrt(x)         ( Kvadratrot )
```

For eksempel,  $\text{sqrt}(M)$  der  $M$  er en matrise, gir en ny matrise der hvert element er lik kvadratrotten av det korresponderende element i  $M$ .

## 2.5 Uvanlige matriseoperasjoner.

Divisjonsoperatorene / og \ kan anvendes på matriser.

Se Pratap, side 58

$A \setminus B$  blir lik  $\text{inv}(A) * B$  der  $\text{inv}(A)$  er den inverse matrisen til A.

$A/B$  blir lik  $A * \text{inv}(B)$  der  $\text{inv}(B)$  er den inverse matrisen til B.

Vanligvis kan de bare anvendes når de inverse matrisene eksisterer og matriseproduktene har mening. ( Se avsnitt 2.6 for annen type bruk ). Det er viktig å være klar over at  $A \setminus B$  beregner  $\text{inv}(A) * B$  raskere enn om vi først hadde beregnet  $\text{inv}(A)$  og deretter utført matrise-multiplikasjon. Det samme gjelder for  $A/B$ .

## 2.6 Løsning av likninger.

Se Pratap,  
side 117-118

Anta vi skal løse likningssystemet  $2 b_1 + 5 b_2 = 12$   
 $3 b_1 + 4 b_2 = 11$

Koeffisientmatrisen betegnes X og vektoren som representerer høyresiden, betegnes y. Likningssystemet løses slik:

```
>> X = [ 2 5  
        3 4 ];  
>> y = [ 12  
        11 ];  
>> b = X \ y;  
>> b1 = b(1)  
>> b2 = b(2)
```

Vanligvis legges flere kommandoer som skal utføres, på en fil.

b blir en kolonnevektor der 1. og 2. element inneholder verdiene til  $b_1$  og  $b_2$ .

*Merknad:*  $A \setminus b$  kan anvendes også når likningssystemet ikke har noen løsning.

Da bestemmes en best mulig løsning ved minste kvadraters metode.

## 2.7 Flere matriseoperasjoner.

Her er en oversikt over noen av de mest brukte matrisefunksjonene:

det(A)	Beregner determinanten
inv(A)	Beregner den inverse matrisen
A'	Beregner A transponert
eig(A)	Beregner egenverdiene
[V,E] = eig(A)	Beregner egenvektorer og egenverdier

Se Pratap, side  
71 -73 og 119-120

Når den siste kommandoen utføres, vil V og E bli to matriser som inneholder henholdsvis egenvektorene og egenverdiene til A. Kolonnene i V er egenvektorene og de korresponderende egenverdiene ligger på diagonalen i E. ( Vi kan gjerne bruke andre navn på matrisene V og E )

## 2.8 Definisjon av nullmatriser, enhetsmatriser osv.

MATLAB har mange kommandoer som lager spesielle matriser som brukes ofte. Her er noen eksempler:

<code>eye(m,n)</code>	lager en m x n matrise med 1-ere på hoveddiagonalen.
<code>zeros(m,n)</code>	lager en m x n matrise der alle elementene er null.
<code>ones(m,n)</code>	lager en m x n matrise der alle elementene er lik 1.
<code>rand(m,n)</code>	lager en m x n matrise der elementene er uniformt tilfeldig fordelte tall mellom 0 og 1.
<code>randn(m,n)</code>	lager en m x n matrise der elementene er normalfordelte tall.

## 3. Plotting av 2-dimensjonale grafer.

Vi illustrerer dette temaet ved følgende program:

```
t = 0 : 0.1 : 4;  
y = 1 + exp( - 0.05 * t.^2);  
plot(t, y)
```

Første linje lager en vektor med t-verdier. Andre linje lager en vektor som inneholder de korresponderende funksjonsverdiene. Legg merke til at den elementvise operasjonen `.^` brukes.

Vi vil nå gi en oversikt over noen av de mange mulighetene når det gjelder å forandre egenskapene til figurene som lages. Til dels vil vi gjøre dette ved kommandoer og til dels ved å bruke menyen i figurvinduet.

### 3.1 Tekst og lengdeskalaer på aksene.

Se Pratap, side 160-161
-------------------------

<code>axis([xmin xmax ymin ymax])</code>	Angir lengdeskalaer på aksene.
<code>xlabel('tekst')</code>	Lager tekst på x-aksen
<code>ylabel('tekst')</code>	Lager tekst på y-aksen
<code>title('Overskrift')</code>	Lager overskrift
<code>text(xpos, ypos, 'text')</code>	Plasserer en tekst på plottet som starter i den angitte posisjon. Posisjonen angis i aksekoordinater.
<code>gtext('tekst')</code>	Når denne kommandoen brukes, så kommer et aksekors tilsyne. Beveg aksekorset dit teksten skal starte og trykk på Enter.

Vi illustrerer disse kommandoene ved å utvide det foregående program:

```
t = 0 : 0.1 : 4;
y = 1 + exp( - 0.05 * t.^2);
plot(t, y)
axis( [ 0 4 0 3 ] )
xlabel('t-verdi' )
ylabel('y-verdi' )
title('Utvikling av . . . ' )
text( 2, 2.5, 'Grafen laget 12.12.01' )
gtext('Grafen er laget ved MATLAB')
```

Se Pratap, side 160

### 3.2 Linjetype, farger og markørtype.

Ved å legge til en parameter i kommandoen plot, så kan linjetype, farger og markørtype varieres. Eksempler:

```
plot(x, y, 'o')      gir en prikket kurve. ( Andre muligheter er '-.' og '--' )
plot(x, y, 'b')      gir en blå kurve. ( Andre muligheter er 'r', 'g' og 'y' )
plot(x, y, 'o')      gir en kurve der hvert datapunkt er markert med o.
                    ( Andre muligheter er '+' og '*' )
plot(x, y, 'b -.')    gir en blå kurve som er tegnet med strek-prikk.
```

Vi presiser at det er mange flere muligheter enn dette.

### 3.3 Flere kurver i samme plott.

Se Pratap, side 163-164

Anta x og y er vektorer med datapunkter som skal plottes mot hverandre.

Anta u og v er vektorer med datapunkter som også skal plottes mot hverandre.

Vi skal beskrive to måter å framstille de tilhørende kurver i samme plot.

Metode 1: `plot(x, y, u, v)`

Metode 2: `plot(x, y, 'r')`

`hold on` % medfører at neste graf kommer

`plot(u, v, 'g')` % i samme plott

`hold off`

Hvis den siste kommandoen (`hold off`) ikke er med, så vil alle grafer som plottes etter hvert, legges i samme figur. Kommandoen `legend` lager en boks på plottet som tilordner en tekst til hver graf på figuren. Før teksten til hver graf angir MATLAB strektype og farge på grafen. MATLAB plasserer boksen der det er best plass på plottet.

Her er et eksempel:

```
x = 0: 0.1 :1;
plot(x,exp(x), 'r')
hold on
plot(x,exp(- x), 'b :')
hold on
plot( x, x.^2, 'g --' )
legend( 'Kurve 1', 'Kurve 2', 'Kurve 3' )
hold on
```

Se Pratap, side 160 og 166

### 3.4 Flere plott i samme vindu.

Kommandoen `subplot(m,n,j)` gir at det opprettes  $m \times n$  forskjellige plottevinduer på skjermen. Den siste parameteren  $j$  angir hvilket av vinduene vi skal plote i.

Her er et programeksempel:

```
x = 0: 0.1 :1;
subplot(2,2,1)
plot(x, exp(x))
subplot(2,2,2)
plot(x, exp(- x))
subplot(2,2,3)
plot( x, x.^2 )
subplot(2,2,4)
plot(x, x.*exp(x))
```

### 3.5 Utskrift og kopiering av figurer.

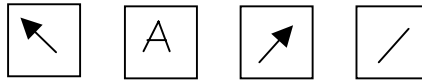
Vi kan få utskrift av figurer ved å klikke på print på menyen file i figurvinduet. Når vi ønsker å få figuren inn i et annet dokument, kan det være lurt å bruke export som også ligger på menyen file. Da kan vi velge lagringsformat. Et fint alternativ er jpeg-format ( alternativt, jpg-format ). Da blir filen lagret som et bilde. Bilde kan da settes inn for eksempel i et Word-dokument ved å bruke menyvalget sett inn bilde i Word. Dessuten kan bilde redigeres, for eksempel ved bruk av Microsoft Photo Editor. En ting som ofte er ønskelig er å fjerne deler av rammen rundt figuren. Hvis vi ikke ønsker rammen i det hele tatt, så kan kommandoen `set(gca,'Box','off')` anvendes. ( `gca` betegner `get current axes` )

### 3.6 Editering i figurvinduet.

Ved å bruke menyen i figurvinduet får vi tilgang på en rekke muligheter.

Vi vil beskrive noen av mulighetene.

På menyen er det fire knapper



Den første knappen slår redigeringsmodus av og på.

Ved å klikke på den andre knappen kan vi lage tekster på plottet.

( Klikk på knappen, klikk der teksten skal begynne og skriv )

Ved å klikke på den tredje knappen kan vi lage piler på plottet.

( Klikk på knappen, klikk der pilen skal begynne og klikk der pilen skal slutte )

Ved å klikke på den siste knappen kan vi lage streker på plottet.

( Klikk på knappen, klikk der streken skal begynne og klikk der streken skal slutte )

Anta redigeringsmodus er på:

- Objekter ( tekster, piler, streker ) kan fjernes ved å venstre-klikke på objektet og deretter klikke på delete-tasten.
- Objekter kan redigeres ved å høyre-klikke på objektet. Da får vi tilgang på en rekke spennende redigeringsmuligheter.
- Objekter kan flyttes ved først å venstre-klikke på objektet og deretter dra objektet bortover ved bruk av en av de tre markørene nederst til høyre på objektet. (Markørene kommer tilsyne når vi klikker på objektet. )

### 3.7 Generelle editeringskommandoer.

Vi kan navngi tekster og kurver vi oppretter.

Kommandoene set og get gir tilgang på egenskaper til objekter og oversikt over valgmuligheter. Vi vil belyse dette ved et programeksempel:

Se Pratap, side 190 - 196

```
x = 0:0.01:1;  
graf1 = plot(x, sqrt(x))  
tekst1 = title('Kvadratrotten av x')
```

Anta dette programmet er utført i kommandovinduet.

Utfør deretter kommandoene `get(graf1)` og `get(tekst1)` . Da får vi oversikt over alle egenskapene til kurve og titteltekst ( de har default-verdier i utgangspunktet )

Utfør deretter kommandoene `set(graf1)` og `set(tekst1)` . Da får vi oversikt over alle valgmulighetene. Her er et programeksempel som viser hvordan egenskaper settes:

```
x = 0 : 0.01 : 1;  
graf1 = plot(x, sqrt(x))  
tekst1 = title( 'Kvadratrotten av x' )  
set( graf1, 'Color', 'r', 'LineWidth', 2)  
set( tekst1, 'Color', 'y', 'FontSize', 18)
```

Farger kan også blandes ut fra basisfargene. Blandingsforholdet angis ved en radvektor av lengde 3 der hvert element er et tall mellom 0 og 1. De tre elementene i vektoren

bestemmer hvor mye av fargene rød, grønn og blå som skal inngå. Her er et eksempel på bruk: `set( tekst1, 'Color',[ 0.3 0.7 0.8 ])`

Vi inkluderer et eksempel til som viser hvor mye forskjellig vi kan få til:

```
x = [ 0.2  0.4  0.6  0.8  1 ];
y = [ 0.3  0.7  0.4  0.6  0.8 ];

kurve = plot( x, y, 'o');
axis( [ 0  1.2  0  1.2] );
set( kurve, 'MarkerSize', 50, 'MarkerEdgeColor','g');
set( kurve, 'MarkerFaceColor','r');
hold on

kurve2 = plot(x, y,'o');
set( kurve2, 'MarkerFaceColor','g');
set( kurve2, 'MarkerSize', 30, 'MarkerEdgeColor','r');
hold off
```

I dette eksemplet blir hvert datapunkt angitt med markøren o. Vi bestemmer størrelse, kantfarge på markøren og farge på flaten av markøren ved bruk av 'MarkerSize', 'MarkerEdgeColor' and 'MarkerFaceColor'. Først tegnes store røde markører med kantfarge grønn. Deretter tegnes de samme punktene en gang til med mindre markører som er grønne og har kantfarge rød. Her er et enda ”villere” eksempel ( en utvidet versjon av foregående eksempel ):

```
x = [ 0.2  0.4  0.6  0.8  1 ];
y = [ 0.3  0.7  0.4  0.6  0.8 ];
kurve = plot(x, y, 'o');
axis( [ 0  1.2  0  1.2] );
set( kurve, 'MarkerSize', 50, 'MarkerEdgeColor', 'g');
set( kurve, 'MarkerFaceColor', 'r');
hold on
kurve2 = plot(x, y, 'o');
set( kurve2, 'MarkerFaceColor', 'g');
set( kurve2, 'MarkerSize', 30, 'MarkerEdgeColor', 'r');
hold off

% For-løkke som skrur de minste markørene av og på:
for i = 1:5
    pause(2);           % Lager pause i to sekunder.
    set(kurve2, 'Visible', 'off');
    pause(2);           % Lager pause i to sekunder.
    set(kurve2, 'Visible', 'on');
end
```

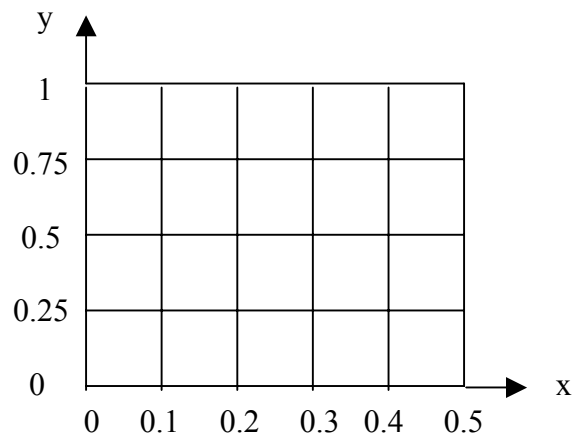
## 4. Tre-dimensjonale plot og diverse plottemuligheter.

Anta vi har et rektangulært grid av punkter i xy-planet med tilhørende z-verdier. Anta x-verdiene og y-verdiene til hvert punkt og de korresponderende z-verdiene er lagret i tre like store matriser som for eksempel er betegnet xmat, ymat og zmat. Da vil kommandoen `surf( xmat, ymat, zmat )` tegne en tre-dimensjonal flate gjennom punktene.

Se Pratap, side 173-189

### 4.1 Et eksempel.

Anta funksjonen  $f(x, y) = x * (0.5 - x) * \exp(-y)$  skal plottes på området gitt ved  $0 < x < 0.5$  og  $0 < y < 1$ . Vi bruker først et lite antall nodepunkter. Da er det lettere å forstå framgangsmåten.



Vi lagrer x-verdiene og y-verdiene i to vektorer

```
x = [ 0  0.1  0.2  0.3  0.4  0.5 ]  
y = [ 0  0.25  0.5  0.75  1 ]
```

Deretter må vi lage matriser som inneholder x-verdiene og y-verdiene til hvert nodepunkt. Heldigvis kan vi bruke kommandoen `meshgrid` til dette: Kommandoen

```
[xmat, ymat] = meshgrid(x,y);
```

gir som resultat matrisene

```
xmat = [ 0  0.1  0.2  0.3  0.4  0.5  
         0  0.1  0.2  0.3  0.4  0.5  
         0  0.1  0.2  0.3  0.4  0.5  
         0  0.1  0.2  0.3  0.4  0.5  
         0  0.1  0.2  0.3  0.4  0.5 ]  
ymat = [ 0  0  0  0  
         0.25  0.25  0.25  0.25  
         0.5  0.5  0.5  0.5  
         0.75  0.75  0.75  0.75  
         1  1  1  1 ]
```

Deretter må vi beregne matrisen som inneholder z-verdiene ved elementvis matrisemultiplikasjon:

```
zmat = xmat.*( 0.5 - xmat).* exp(- ymat);
```

Deretter plottes flaten ved kommandoen `surf(xmat, ymat, zmat)`.

Her er et program for samme eksempel med flere nodepunkter:

```
x = 0 : 0.05 :0.5;
y = 0 : 0.05 :1;
[xmat, ymat] = meshgrid(x,y)
zmat = xmat.*( 0.5 - xmat).* exp(- ymat);
surf( xmat, ymat , zmat )
```

I figurvinduet kan vi trykke på roterknappen. Da kan vi klikke på plottet, holde museknappen nede og dra plottet rundt. Vi kan da se det fra den vinkel vi ønsker. Tekst på aksene osv. gjøres på samme måte som for 2-dimensjonale plott.

Her er en utvidet versjon av foregående eksempel:

```
x = 0 : 0.05 :0.5;
y = 0 : 0.05 :1;
[xmat, ymat] = meshgrid(x,y)
zmat = xmat.*( 0.5 - xmat).* exp(- ymat);
surf( xmat, ymat , zmat );
axis( [ 0 0.5 0 1 0 0.1 ] ),
xlabel( 'x' );
ylabel( 'y' );
```

Ved å legge til kommandoen `shading interp` glatter vi ut flaten (en av mange muligheter).

#### 4.2 Konturplott.

Vi vil også forklare hvordan konturplott kan lages til eksemplet  $f(x, y) = x*y$ :

```
x = -3:0.2:3;
y = -3:0.2:3;
[xmat, ymat] = meshgrid(x, y);
zmat = xmat.*ymat;
[C, H] = contour(xmat, ymat, zmat);
clabel(C,H) % Setter tall på kurvene ( valgfritt )
```

Se Pratap, side 172

C og H er en matrise og en vektor som inneholder den informasjonen clabel trenger.

### 4.3 Diverse plottemuligheter.

Det er en rekke andre kommandoer for å lage plott i 2- og 3- dimensjoner. Her vil vi kort beskrivelse noen få muligheter.

<u>Kommando</u>	<u>Kommentar</u>	Se Pratap, side 167-172 og 181 -185
<code>fill( x, y, 'r' )</code>	Lager et fargelagt polygon der koordinatene til hjørnene er gitt ved vektorene x og y. Vi har valgt fargen rød.	
<code>hist(x)</code>	Lager histogram over dataene i vektoren x.	
<code>hist(x, n)</code>	Lager histogram over dataene i x med n stolper.	
<code>plot3(x, y, z)</code>	Plotter en tre-dimensjonal kurve der koordinatene til punktene er gitt ved vektorene x, y og z.	

Det er dessuten et stort mangfold av muligheter til å gi tekstobjekter, linjeobjekter osv. de ønskede egenskaper. ( Figurene er inndelt i grafiske objekter som kan tillegges egenskaper )

Se Pratap, side 190 - 201

## 5. Lagre og hente data.

### 5.1 MATLAB-format.

Se Pratap, side 74

Enkel lagring av data på fil gjøres med `save`

```
>> save datafil y X b;
```

Vi lagrer her variablene y, X og b på filen datafil.mat. Endelsen .mat settes automatisk på, og filen får et eget matlab-format som ikke kan leses av andre programmer. For enkel innlesing av data fra fil bruker vi `load`. Dersom vi har lagret variabler i Matlab-format på filen datafil.mat kan vi skrive

```
>> load datafil;
```

og alle variable leses inn og opprettes med samme navn som de ble lagret med.

### 5.2 Ren tekst.

Vi kan også lagre og lese rene tekst-filer (ASCII-format). Slike filer kan også leses av andre programmer. Vi lagrer data som ren tekst ved

```
>> save -ascii datafil.txt y X b;
```

Her lagres variablene på filen datafil.txt (endelsen må du nå selv putte på). Det er ikke lurt å lagre flere variable på samme tekst-fil, selv om det er mulig. Det er nemlig slik at dersom slike filer skal kunne leses av MATLAB igjen, må du kun legge en matrise på hver fil. Når du leser inn igjen slike filer får matrisen i MATLAB automatisk samme navn som filen (bortsett fra endelsen).

```
>> load Xdata.txt ;
>> load ydata.txt ;
>> load bdata.txt ;
```

Vi får dermed variablene Xdata, ydata og bdata opprettet i Matlabs arbeidsrom. Disse kan vi så døpe om etter behag.

### 5.3 Andre formater.

MATLAB kan importere data fra mange andre formater, for eksempel direkte fra Excel-filer. Skriv help fileformats for mer hjelp om dette.

For mer avansert input/output, se Pratap, side 104

## 6. Mere matrisemanipulasjon.

### 6.1 Kort repetisjon.

Vi henter ut en rad fra en matrise slik

```
>> rad2 = X(2, :);
```

der vi hentet ut rad 2. Tilsvarende for kolonner

```
>> kolonne3 = X(:,3);
```

Vi kan skjøte matriser dersom dimensjonene tillater det. Anta matrise X1 er (10x4) og X2 er (7x4). Da kan disse legges under hverandre fordi de samme antall kolonner

```
>> X = [X1 ; X2];
```

slik at X blir 17x4. Tilsvarende, dersom antall rader hadde vært like kunne vi ha skjøtet de sammen ved å legge de ved siden av hverandre slik  $X = [X1 \ X2]$ ;

### 6.2 Logiske vektorer og matriser.

Vi kan lage logiske uttrykk med matriser, og får da generert det vi kaller logiske matriser, som bare har verdiene 0 eller 1. Anta matrisen X inneholder både positive og negative tall. Vi vil vite hvor de positive tallene ligger i matrisen, og skriver

```
>> positive = ( X > 0 );
```

positive blir da en matrise med samme dimensjoner som X, og med 1 der X er positiv og 0 ellers. Parentesene er ikke strengt tatt nødvendige, men anbefales for å øke leseligheten! En funksjon som ofte brukes sammen med logiske matriser/vektorer er find. Denne returnerer indeksene i en logisk matrise/vektor der verdien er 1. Et eksempel belyser dette:

Anta vi har en serie med data i vektoren  $x$ . Vi vil finne største verdi i  $x$  og hvor den ligger. Største verdi i en vektor finner vi ved kommandoen `max`

```
>> xMax = max(x);
```

Vi lager så en logisk vektor som angir hvor største verdi ligger

```
>> xLogisk = ( x == xMax );
```

og indeksene til denne største verdien er gitt ved

```
>> xMaxIndeks = find( xLogisk );
```

Vi kunne ha gjort alt på en linje slik

```
>> xMaxIndeks = find( x == max(x) );
```

men slik kompakt notasjon er ikke alltid like lett å lese for andre!

### 6.3 Vektorindeksering.

Vi kan på enkelt vis trekke ut en vilkårlig delmengde av elementer fra en matrise eller vektor. Gitt vektoren

```
>> indeks = [1 3 4 7 2];
```

og anta  $y$  er en  $(10 \times 1)$  vektor og  $X$  er en  $(10 \times 4)$  matrise. Vi kan da skrive

```
>> ydel = y(indeks);  
>> xdel = X(indeks, :);
```

der  $y_{del}$  består av element 1, 3, 4, 7 og 2 av  $y$ , og  $x_{del}$  blir en matrise med radene 1,3,4,7 og 2 fra  $X$ .

## 7. Enkel programmering.

I Matlab kan vi lage to typer ”programmer”, script eller funksjoner. Vi har allerede sett bruk av script, som bare er en samling Matlab-kommandoer på en fil. Når vi skriver filens navn i kommandovinduet (vi sier vi ”kaller” scriptet) utføres kommandoene i filen sekvensielt som om de ble tastet inn etter tur i kommandovinduet. Vi vil se litt mer på funksjoner om litt.

### 7.1 Kontrollstrukturer.

Se Prata, kapittel 4
----------------------

Ofte vil vi i et program ha repetert noen kommandoer et gitt antall ganger. Til dette kan vi typisk bruke **for**-løkker. En **for**-løkke i Matlab ser slik ut

```
for i = 1:10  
    x = 2*i;  
end
```

Alt som står mellom **for** `i = 1:10` og **end** blir nå utført 10 ganger. Variabelen `i` kalles tellevariabelen, og den får økende verdi (1, 2, 3, ..., 10) for hver runde løkka går. Dersom vi vil at `i` skal øke med steg på bare 0.5 måtte vi ha skrevet `for i = 1:0.5:10`. Du velger fritt hvor tellingen skal starte og slutte. **for**-løkker brukes typisk til å søke gjennom rader/kolonner i matriser/vektorer.

Ofte vil vi ha testet om en eller flere betingelser er oppfylt. Til dette bruker vi typisk en **if**-setning. Vi kan ha **if**-setninger med eller uten **else**-ledd

```
if a<5
    b=a^2;
end
```

eller

```
if a<5
    b=a^2;
else
    b=a^1.5;
end
```

Vi kan også nøste flere **if**-setninger etter hverandre ved å bruke **elseif**

```
if a<5
    b=a^2;
elseif a>10
    b=a;
else
    b=a^1.5;
end
```

En slik nøstet if-setning kan avsluttes med eller uten et **else**-ledd.

## 7.2 Utskrift og innlesing fra skjerm.

For enkelt å hente inn data fra skjermen til et program kan vi bruke `input`. Du oppgir da en lede-tekst som skrives ut:

```
>> a=input('Oppgi et tall: ');
```

Tekster skal alltid stå mellom enkle apostrofer ('tekst'). Du kan oppgi matriser av vilkårlig dimensjon til `input`. For å skrive tekster ut til skjerm bruker vi `disp`

```
>> disp(' Dette er en tekst');
```

Dersom du vil putte inn et tall må du først gjøre om tallet til tekst (`int2str()` for heltall, `num2str()` for desimaltall) og deretter skjøte dette til den øvrige teksten. MATLAB skjøter strenger (tekster) akkurat som vektorer:

```
>> disp(['Resultatet ble ' num2str(svar)]);
```

der strengen 'Resultatet ble ' skjøtes sammen med strengen som `num2str(svar)` lager. Dersom svar har verdien 4 blir strengen ' 4 '.

## 8. Egne funksjoner.

Etter hvert som man blir en mer dreven bruker av MATLAB vil man oppdage nytten av å kunne lage sine egne funksjoner. En funksjon i MATLAB tilsvarer det som heter funksjoner, prosedyrer eller metoder i andre programmerings-språk. Kort sagt er funksjoner *innkapsling av operasjoner*.

En funksjon skiller seg fra et script ved at den alltid starter med nøkkelordet `function`. En funksjon kan også ta en eller flere *parametre*, og kan returnere en eller flere *returvariable*. La oss se på et eksempel. Vi vil lage en funksjon som kaster N terninger, dvs. trekker N tilfeldige heltall fra 1 til 6. I editor-vinduet åpner vi en ny, tom, m-fil, og skriver

```
function oyne = terning(N)
% Dette er en funksjon som simulerer kast med N terninger.
% Resultatene returneres i en kolonne-vektor
oyne = ceil(rand(N,1)*6);
```

der vi først trekker N tilfeldige tall mellom 0 og 1 ( med `rand(N,1)` ), deretter multipliseres alle tallene med 6 slik at vi får tilfeldige tall mellom 0 og 6, og til slutt rundes alle tall oppover til nærmeste heltall ( med `ceil` ).

Legg merke til:

<code>ceil(a)</code> runder alltid oppover <code>floor(a)</code> runder alltid nedover <code>round(a)</code> runder til nærmeste (opp eller ned)
--

- En funksjon begynner alltid med nøkkelordet `function`
- Returvariabelen er i dette tilfellet `oyne`. Vi kan ha mange returvariabler. Disse omslutes da med `[]` og skilles med komma.
- Funksjonens navn kommer rett etter `=`.
- **NB! Filen må ha samme navn som funksjonen** (her `terning.m`).
- Parametere listes mellom parentesene etter navnet
- Parameter er i dette tilfellet `N`. Vi kan ha mange parametre. Disse skilles da med komma.

Når denne funksjonen er laget kan vi bruke den som en hvilken som helst annen MATLAB kommando:

```
>> yatzyTerninger = terning(5);
```

og variabelen `yatzyTerninger` blir en vektor med 5 tilfeldige tall fra 1 til 6.

Når du lager et script, slik vi har gjort til nå, blir alle variable som opprettes underveis værende i MATLABs arbeidsrom. I en funksjon er ikke dette tilfelle. Alle variable i en funksjon er i utgangspunktet lokale, dvs. de opprettes når funksjonen kalles, og forsvinner så

fort den er ferdig. Det er bare returverdiene som ”lever videre”. På denne måten kan du ”pakke inn” operasjoner i en funksjon, og deretter bruke denne uten å ta hensyn til hva som foregår inne i funksjonen.

## 9. Hjelpemulighetene i MATLAB.

Se Pratap, avsnitt 3.4

Dersom du er usikker på bruk av en kommando kan du skrive `help` og kommandonavnet direkte i kommandovinduet. Du får da en kort beskrivelse av kommandoen. Når du lager egne funksjoner er det linjene med kommentarer som kommer direkte etter første linje som vises når noen skriver `help` og funksjonens navn.

Ofte står vi overfor et problem, og lurer på om MATLAB har en funksjon som kan hjelpe oss. Da kan vi skrive `lookfor` og stikkord i kommandovinduet. MATLAB lister da alle kommandoer som trolig har noe med stikkordet å gjøre.

For en mer utfyllende hjelp skriver du `helpdesk` i kommandovinduet, og får opp et eget hjelpe-vindu der du kan søke etter funksjoner, temaer og lignende.

## 10. Et utvalg av MATLABs muligheter for viderekommende.

MATLAB har mange muligheter utover det som nevnes i dette korte skrevet. Vi vil nevne noen av de viktigste her.

### 10.1 Verktøykasser (toolboxes).

Vi har så langt ikke benyttet noen av de mange verktøykassene som finnes for MATLAB. Slike verktøykasser består av en samling funksjoner som har vist seg nyttige for et bestemt formål. Disse kjøper du ekstra fra leverandøren av MATLAB, og de krever alle at du har selve MATLAB fra før. Det finnes etter hvert mange(!) slike verktøykasser, og noen vi har hatt glede av er

- Optimization toolbox (optimering av alle slag)
- Statistics toolbox (fordelinger, clustring, PCA, osv...)
- Image analysis (bildebehandlign og analyse...)
- Signal processing (signalbehandling...)
- Compiler (mer om denne nedenfor...)
- Symbolic math (symbolsk matematikk)
- Simulink (stort grafisk system for simulering med differensiallikninger)

## *10.2 Grensesnitt mot andre språk.*

MATLAB-programmer kan ”snakke med” programmer i andre språk! Både C, Fortran og Java er språk som MATLAB kommuniserer med.

Grensesnittet mot C og Fortran er først og fremst laget for a) å kunne flytte tunge beregninger over på disse raske språkene, eller b) for først å kunne utvikle programmer i MATLAB og deretter oversette de til egne program som ikke trenger MATLAB for å kjøres.

Selv om MATLAB kjører ganske fort, er det alltid mulig å øke hastigheten med å oversette til C eller Fortran. Til dette trenger du MATLAB Compiler, som er listet blant verktøykassene over. I tillegg må du også ha en kompilator for disse språkene innstallert på din maskin. Mange bruker MATLAB under utvikling av programvare, men det ferdige produktet skal selges som en selvstendig pakke (”stand-alone”). Da trenger du også Compiler, men må i tillegg ha noen biblioteker av ferdige C eller Fortran-funksjoner som gjør at sluttproduktet blir helt uavhengig av MATLAB. En slik oversatt kode har flere fordeler: Du trenger ikke selve MATLAB for å kjøre de, de kjører raskt og selve koden er uleselig for kundene som kjøper produktet.

Grensesnittet mot Java er det siste som har kommet, og skiller seg noe fra de andre. Du kan nemlig kalle Java-klasser direkte fra MATLAB uten noen kompilering! Dette betyr at du kan lage grafiske grensesnitt i Java, innlesing fra fil via Java, eller aller mest typisk, kjøre programmer over Internett direkte fra MATLAB (via Java).

## *10.3 Grafisk grensesnitt.*

Du kan lage egne grafiske grensesnitt i MATLAB. Med grafiske grensesnitt mener vi paneler av knapper og hendler som gjør at man ikke trenger skrive kommandoer i kommandovinduet. Flere av verktøykassene bruker slike grensesnitt (for eksempel signal prosessing). Vi anbefaler bruk av verktøyet guide til å bygge egne grensesnitt. Les mer om guide i helpdesk. Nå som MATLAB ”snakker” Java er kanskje dette ikke like aktuelt som det en gang var...

## *10.4 Ressurser på Internett.*

Siden MATLAB er svært utbredt over hele verden ligger det fantastisk mye ferdig kode på Internett. Selgerne av MATLAB, MathWorks (<http://www.mathworks.com/>), har egne biblioteker med kode som folk har laget. Du finner også mye gratis MATLAB-kode under MathTools sine hjemmesider (<http://www.mathtools.net/>). På NLH har vi også en side som du kan ha glede av å kikke på, nemlig sidene til PROBE (Programvare for beregning og visualisering, <http://arken.nlh.no/matlab/>).