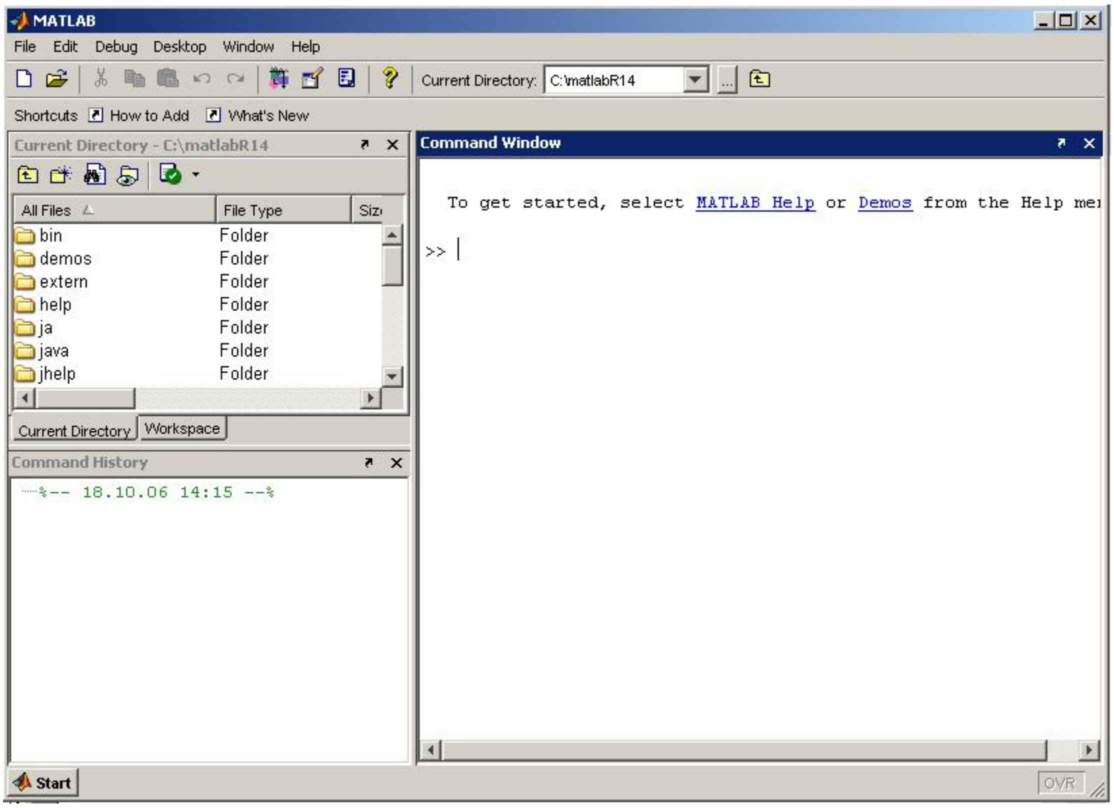
En kort innføring i Matlab

Halvor Aarnes /BIO2110 H2006. S.E.& O. 15-11-2007

Innholdsfortegnelse

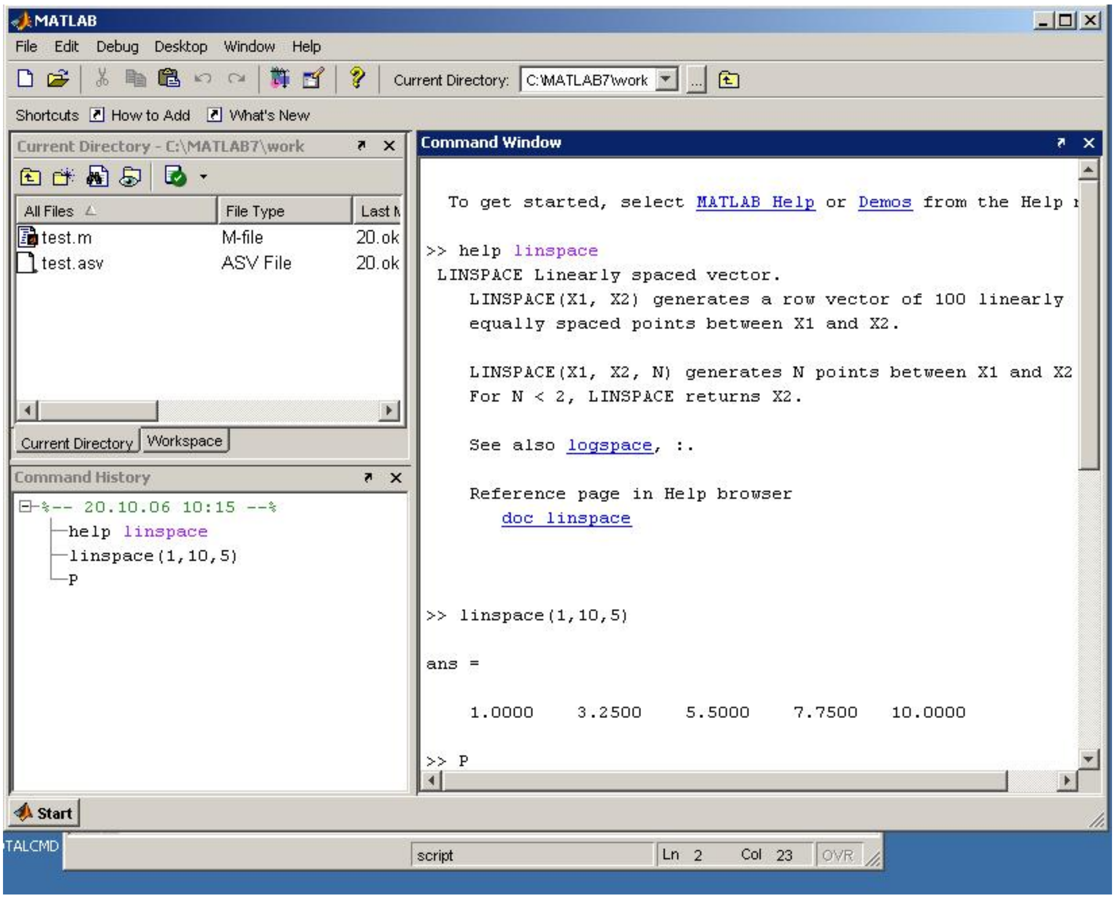
Matlab ("Matrix Laboratory") er et regneprogram som kan brukes til beregninger, programmering, numerisk løsning av differensialligninger samt modellering. Det er et kostbart program, og et godt alternativ er programmet R.

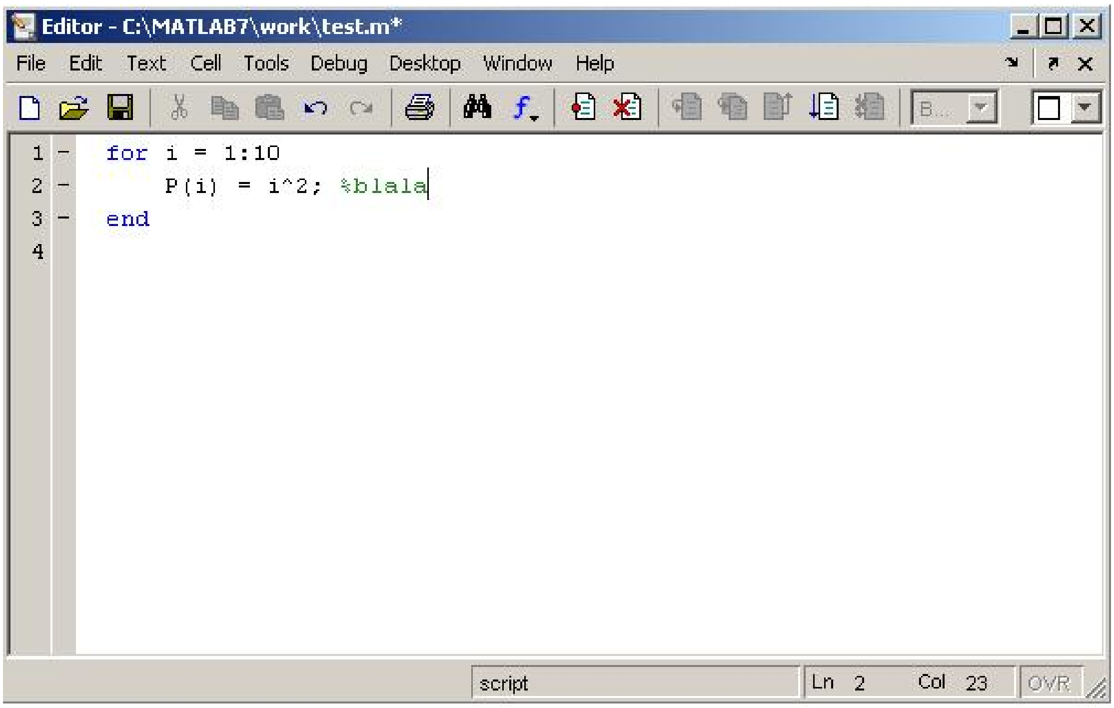


Default-vindu i Matlab.

Command Window - Hovedvindu hvor man setter inn variable og herfra kan programmer kjøres.
Current Directory - Viser filene i direktoriet
Workspace window - Innholder informasjon om variable.
Figure Window - Åpnes automatisk når grafikk vises
Command History Window -
Editor Window - brukes til å skrive og editere programmer
Launch Pad Window -
Help Window - Hjelp informasjon
Man får hjelp ved kommandoen **help**. Vil du ikke ha alle de tomme linjene som matlab lager bruk kommandoen **>>format compact**.

Kommandoen **>> help** gir en oversikt over alle hjelp
>> help elmat gir oversikt over matrisekommandoer
>>help specfun gir oversikt over spesielle mattekommandoer. Andre hjelpkommandoer er:
help funfun, help polyfun, help plotxy, help plot xyz, help lang, help matfun, help ops.





Script-filer lages i Editor/debugger-vindu. Man starter med File og New og velger M-file. En ny kommandolinje kommer for hvert Enter. Filene kan lagres vis Save As..En fil kan eksekveres via kommandovindu eller direkte i Editorvindu ved å klikke på Run-ikonet.

Skjermkommandoer er **fprintf** og **disp**.

Kommandovinduet

Prompten er **>>** som viser at programmet venter på en kommando. En kommando utføres ved å trykke på Enter-knappen. Tidligere kommandoer finner man igjen ved å bruke piltastene opp (↑)eller ned (↓).
Semikolon (;) etter kommando gjør at kommandoen ikke blir utført.
Prosent (**%**) ved starten av linje viser at linjen brukes til kommentarer. Kommentarer brukes mye når programmer skal skrives.
Kommandoen **clc** etterfulgt av Enter fjerner alt fra kommandovinduet.

Matlab som kalkulator

```
>> 5+15/16
ans =
    5.9375
>> 6^3/5
ans =
   43.2000
```

Du kan formatere utskriften fra Matlab med kommandoene **format short**, format long, format short e, format bank, format compact.
Det er en rekke innebygde funksjoner:
sqrt(x), exp(x), log(x), abs(x), log10(x), factorial(x), sin(x), cos(x), tan(x), cot(x), round(x)avrunder til nærmeste heltall, fix(x), ceil(x),floor(x), rem(x), sign(x). men også mer kompliserte funksjoner som **gamma**, **besselj** og **erf**. Kommandoen **inf** er uendelig og i er komplekse tall i=sqrt(-1)

For eksempel $e^{3.5}$:

```
>> exp(3.5)
ans =   33.1155
```

Definere konstanter eller et tall:

```
>> x=25
x =
    25
>> b=3.2;
>> x*b
ans =  80
```

Legg merke til forskjellene:

```
>> 6^2/5
ans = 7.2000
```

og

```
>> 6^(2/5)
```

```
ans = 2.0477

>> navn='Mitt navn er Halvor'
navn =
Mitt navn er Halvor
>>
```

Noen variable er forhåndsdefinert: pi, inf, i
Kommandoen **clear** fjerner alle variable fra hukommelsen, men man kan etter clear vise hvilke variable som ønskes fjernet. **who** og **whos** viser hvilke variable som finnes.
Vi vet at cos π/2=0, men setter vi dette inn får vi:

```
>> cos(pi/2)
ans =  6.1232e-017
>> %Skal dette bli riktig må vi bruke symboler
>> cos(sym('pi/2'))
ans =0
>>
```

Kommandoene **sym** og **syms** er beslektet. Fkes. Vil **syms x** være synonymt med **x=sym('x')**.

Kommandoen variabel-presisjons aretmetikk **vpa** kan bestemme hvor mange desimaler du vil ha etter komma.

```
>> vpa('sqrt(2)',50)
ans =
1.4142135623730950488016887242096980785696718753769
>>
>> x=[1:1:10]
x =
  Columns 1 through 9
     1     2     3     4     5     6     7     8     9
  Column 10
    10
x.^2
ans =
  Columns 1 through 9
     1     4     9    16    25    36    49    64    81
  Column 10
   100
>>Husk at vi må bruke .^
```

Tallene fra 1:100 plassert i en radvektor og deretter summert:

```
>> z=[1:1:100];
>> sum(z)
ans =  5050
```

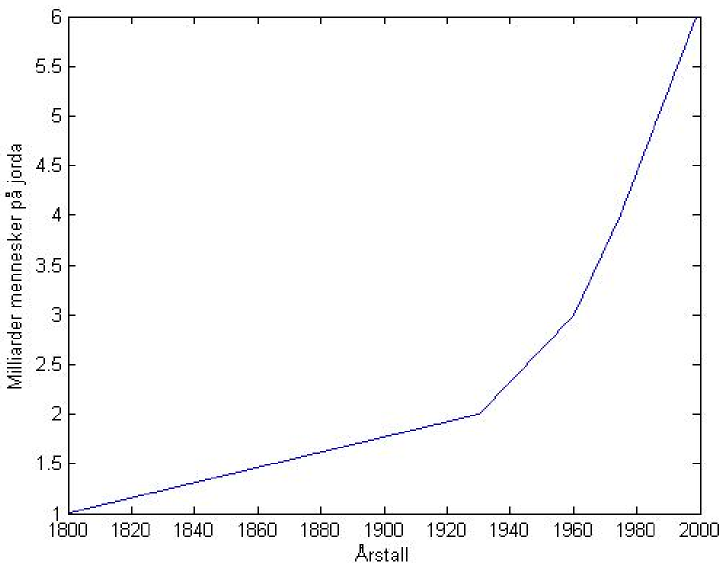
Produktet av alle tallene 1:100

```
>> prod(z)
ans =  9.3326e+157
>
```

Matriser (arrays)

Man kan lage en vektor over befolkningen på jorda i milliarder. Årstall som radvektor og antall milliarder som kolonnevektor. Avslutt linjen med semikolon (;) for å slippe utskrift av tallene.

```
>> year=[1800 1930 1960 1975 1987 1999];
>> pop=[1;2;3;4;5;6];
>> year
year =
  Columns 1 through 4
        1800         1930         1960         1975
  Columns 5 through 6
        1987         1999
>> pop
pop =
     1
     2
     3
     4
     5
     6
>> plot(year,pop)
>> xlabel('Årstall')
>> ylabel('Milliarder mennesker på jorda')
```



Man kan lage en vektor med konstant mellomrom:

```
>> x=[1:1:10]
x =
  Columns 1 through 9
     1     2     3     4     5     6     7     8     9
  Column 10
    10
```

<

Kommandoene **zeros(x,y)** lager matrise med bare 0, **ones(x,y)** matrise med enere, og **eye(n)** brukes til å lage en matrise med 1 i diagonalen. En oversikt over matriseoperasjoner fås med kommandoen **help elmat**.

```
>> A=zeros(5,4)
A =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> B=ones(3,5)
B =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
>> C=eye(6)
C =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
>>
```

Alle variable i matlab uttrykkes som matriser, hvor en skalar har bare ett element, en vektor har bare en rad, og en matrise består av rader og kolonner.
Vektorer og matriser kalles tabeller (arrays)

Tilordner en skalar til x

```
>>x=12.4
x =
   12.4000
```

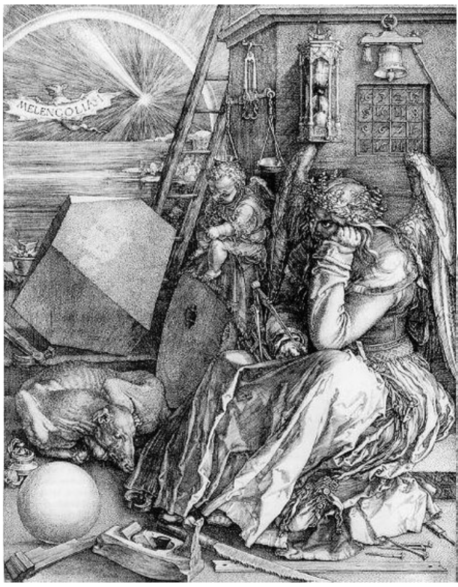
En liggende vektor:

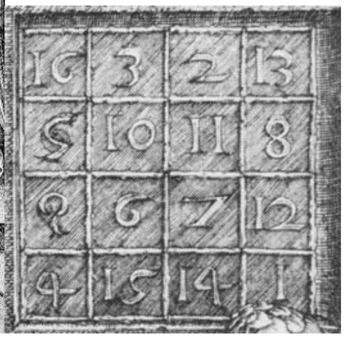
```
>> y=[2 4 6 8]

y =

     2     4     6     8
```

Elementene i en rad atskilles fra neste rad med semikolon ;





En matrise er en rektangulær samling av tall i rader og kolonner. Matriser med bare en kolonne eller en rad kalles en vektor. En 1x1 matrise er bare et tall, en skalar.
Jfr. Albrecht Düreres Melencolia I med et magisk kvadrat. Det finnes ikke noe magisk i kvadratet, men har den interessante egenskapen at tallet 34 går igjen i flere av summeringen.

- Alle hjørnene summeres til 34
- De fire tallene i midten summeres til 34
- 3 og 2 i første rad som vender mot 15 og 14 i fjerde rad summeres til 34
- 5 og 9 i første kolonne som vender mot 8 og 12 i fjerde kolonne summeres til 34
- De fire kvadratene i hvert hjørne adderes til 34
- Summeres kolonnene blir dette 34
- 15+9+2+8 = 34
- Summen av diagonalene blir 34

16	3	2	13
5	10	11	8
2	6	7	12
4	15	14	1

Matlab bruker filutvidelseskoden .m. Programmet er som navnet sier spesialtilpasset til å bruke matriser som objekter.

Vanlig regneoperasjoner addisjon (+) av tall må ha samme dimensjon.
Skal det multipliserer element for element for eksempel ved multiplisering av to vektorer bruk kommandoen **.***

Alle elementene omgis av en hakeparentes [ ] og hver rad i kolonnen atskilles med ;

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

A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1

>> sum(A)

ans =

    34    34    34    34
```

A' transposerer matrisen ved å vende den om diagonalen. Hvis kommandoen brukes på en vektor skifter radvektoren til en kolonnevektor.

```
>> A'

ans =

    16     5     9     4
     3    10     6    15
     2    11     7    14
    13     8    12     1

>> sum(A')'

ans =

    34
    34
    34
    34
```

Tallene i en diagonal i matrisen summeres med:

```
>> diag(A)

ans =
```

```
    16
    10
     7
     1
```

```
>> sum(diag(A))

ans =

    34
```

Et tall i i-rad og j-kolonne har betegnelsen A(i,j)

```
>> 1:10

ans =

     1     2     3     4     5     6     7     8     9    10
```

Kommandoen **magic** lager et magisk kvadrat:

```
>> B=magic(4)

B =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

En 1x1 matrise

```
>> ant_studenter=25

ant_studenter =

    25
```

Concatenering er å koble sammen mindre matriser til større.

Multiplisering av en matrise med en transposert matrise gir en symmetrisk matrise:

```
>> A'*A

ans =

   378   212   206   360
   212   370   368   206
   206   368   370   212
   360   206   212   378
```

Determinanten til en kvadratmatris bestemmes med kommandoen **det(A.)** Determinanten til A = 0 noe som indikerer en **singular matrise**.

```
>> d=det(A)

d =

     0
```

En singular matrise har ingen invers matrise. Egenverdiene til en kvadratmatrise A finnes med kommandoen eig(A). Kommandoen **[U,R]=eig(A)** viser både egenverdier og egenvektorer

```
>> e=eig(A)

e =

   34.0000
    8.0000
   -0.0000
   -8.0000
```

En av egenverdiene =0 er et resultat av singularitet. Største egenverdi er 34, den magiske sum. Dette fordi vektor til alle ones er en egenvektor

```
>> v=ones(4,1)

v =

     1
     1
     1
     1

>> A*v

ans =

    34
    34
    34
    34
```

Når en magisk matrise er skalert på dens magiske sum

```
>> P=A/34

P =

    0.4706    0.0882    0.0588    0.3824
    0.1471    0.2941    0.3235    0.2353
    0.2647    0.1765    0.2059    0.3529
    0.1176    0.4412    0.4118    0.0294
```

som er en dobbel stokastisk matrise hvor alle rader og kolonnesummer er lik 1

Slike matriser representerer transisjon av sannsynlighet i Markov prosesser

```
>> P^5

ans =

    0.2507    0.2495    0.2494    0.2504
    0.2497    0.2501    0.2502    0.2500
    0.2500    0.2498    0.2499    0.2503
    0.2496    0.2506    0.2505    0.2493
```

Når k går mot uendelig går p^k mot 1/4

Koeffisientene i polynomet

```
>> y=x.^3;
```

En identitetsmatriser er en kvadratmatrise hvor alle tall i diagonalen er lik 1 og resten er lik 0. En matrise (A) kan bli multiplisert med identitetsmatrisen I: AI=IA=A. Hvis B er en invers matrise til A vil AB=I. Determinanter er assosiert med kvadratmatriser. Matlab har to måter å dividere matriser: venstredivisjon (\) og høyredivisjon (/)

```
>> A=[16 3 2 13;5 10 11 8;9 6 7 12; 4 15 14 1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> [U,R]=eig(A)%Finner både egenverdier og egenvektorer
U =
  Columns 1 through 3
  -0.50000000000000  -0.81649658092773   0.22360679774998
  -0.50000000000000   0.40824829046386  -0.67082039324994
  -0.50000000000000   0.00000000000000   0.67082039324994
  -0.50000000000000   0.40824829046386  -0.22360679774998
  Column 4
  -0.40824829046386
  -0.00000000000000
  -0.40824829046386
   0.81649658092773
R =
   Columns 1 through 3
  33.99999999999996                  0                  0
                  0   8.00000000000001                  0
                  0                  0   0.00000000000000
                  0                  0                  0
  Column 4
                  0
                  0
                  0
  -8.00000000000000
>>
```

Vil man heller ha skrevet ut egenvektorene som rasjonale tall bruk kommandoen **format rat.** For å komme tilbake til vanlig format avslutt med **format short.**

```
>> format rat
>> [U,R]=eig(A)

U =
    -1/2        -881/1079     646/2889     -881/2158
    -1/2         881/2158    -646/963          *
    -1/2           *          646/963      -881/2158
    -1/2         881/2158    -646/2889      881/1079

R =
     34            0            0            0
      0            8            0            0
      0            0            *            0
      0            0            0           -8
>>format short
```

Egenparene finnes med:

```
>> [U,R]=eig(sym(A))
U =
[ -1,  1,  1, -2]
[  3,  1,  0,  1]
[ -3,  1,  1,  0]
[  1,  1, -2,  1]
R =
[  0,  0,  0,  0]
[  0, 34,  0,  0]
[  0,  0, -8,  0]
[  0,  0,  0,  8]
>>
>> plottools
```

Hvis vi har to matriser A og B kan hvert av elementene i matrisen summeres:

```
>> A=[1 -1 0;2 3 4]
A =
     1    -1     0
     2     3     4
>> B=[5 1 -1;4 4 3]
B =
     5     1    -1
     4     4     3
>> A+B
ans =
     6     0    -1
     6     7     7
```

Vi kan multiplisere hvert element i A med et tall:

```
>> 3*A
ans =
     3    -3     0
     6     9    12
<
```

Skal matrisene multipliseres med hverandre komponentvis bruk .*

```
>> A.*B
ans =
     5    -1     0
     8    12    12
```
Man kan bruke regneoperasjoner på matriser:
```
>> cos(A)
ans =
    0.5403    0.5403    1.0000
   -0.4161   -0.9900   -0.6536
```

Du kan skrive ut deler av en matrise:
```
>> A=[16 3 2 13;5 10 11 8;9 6 7 12; 4 15 14 1];
>> A(1,:)
ans =
    16     3     2    13
>> A(:,2)
ans =
     3
    10
     6
    15
```

Lage egne funksjon

Det går an å lage egne funksjoner og en metode er via funksjonen **inline** eller bedre via **@**.

```
>> f=@(x) x^3
f =
    @(x) x^3
>> f(3)
ans =
    27
>> %For å være sikker på at funksjonen virker på matriser bruk .^og .*

>> fxy=@(x,y) x.^3+y.^3
fxy =
    @(x,y) x.^3+y.^3
>> fxy([1 1],[2 3])
ans =
     9    28
```
Som er verdiene for punktene (1,2) og (1,3)

Script-filer i Editor-vindu

Her lages først en matrise og fprintf viser to tall fra A for hver linje og kommandoen gjentas 5 ganger.

```
x=1:4;
y=log(x)
A=[x;y]
fprintf('Hvis tallet er:%i,logaritmen er:%f\n',A)
```

```
y =
         0    0.6931    1.0986    1.3863
A =
    1.0000    2.0000    3.0000    4.0000
         0    0.6931    1.0986    1.3863
Hvis tallet er:1,logaritmen er:0.000000
Hvis tallet er:2,logaritmen er:0.693147
Hvis tallet er:3,logaritmen er:1.098612
Hvis tallet er:4,logaritmen er:1.386294
```

Kommandoen **fprintf** kan også brukes for å skrive til en fil.

Filer kan importeres og eksporteres for eksempel importere Excel-filer:
Variabel_navn=xlsread('filnavn')
Eller man kan bruke import direkte i matlab.

Løsning av ligninger

Ligning med koeffisienter i matrise A og høyreside er vektoren b.
Løsningen for x i Ax=b. $x=A^{-1}b$. Bruk heller Gauss-eliminering eller \. X=A\b

```
>> C=rand(4)
C =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
>> rank(C)
ans =
     4
>> det(C)
ans =
    0.1155
```

Singulær matrise har rang 2 og determinant = 0

Matriseregning: **.*   .^   ./   .**

Hvis vi har tre ligninger:
5x – 3y + 8z = 9

3x + 7y + 6z = 5
6x + 9y + 4z = 1

Så kan disse uttrykkes som en matrise AX=B hvis A er en **kvadratisk ikke-singulær matrise** og B er en kolonnevektor

```
>> A=[5 -3 8;3 7 6;6 9 4];
>> B=[9;5;1];
>> X=A\B
X =
   -0.3478
   -0.2174
    1.2609
```

Vi kan som et alternativ lage en **utvidet matrise** for ligningssystemet over:

```
>> ab=[5 -3 8 9;3 7 6 5;6 9 4 1]
ab =
     5    -3     8     9
     3     7     6     5
     6     9     4     1
```

Denne matrisen kan omdannes til **trappeform** med kommandoen **rref**(ab) (rrf="reduced row echelon form")

```
>> rref(ab)
ans =
    1.0000         0         0   -0.3478
         0    1.0000         0   -0.2174
         0         0    1.0000    1.2609
>>
```

Matrisen i trappeform ser vi her starter med tre **pivotsøyler** som starter med **pivotelementene** 1. Og vi ser at vi kommer fram til samme sluttresultat som metoden foran:
x= -0.3478, y= -0.2174 og z= 1.2609

Noen ganger kan det være gunstig å foreta **radoperasjoner** manuelt for hver rad i stedet for kommandoen rref.
Det er en rekke funksjoner som kan brukes på matriser:
mean(A); **C=max(A)**; **min(A)**; **sum(A)**; **sqrt(A)**; **median(A)**; **std(A)**; **det(A)**; **dot(x,y)**; **cross(x,y)**; **inv(A)**.

Kommandoen **rand** lager tilfeldige tall mellom 0 og 1
randn lager normalfordelte tall med middel 0 og standardavvik 1.

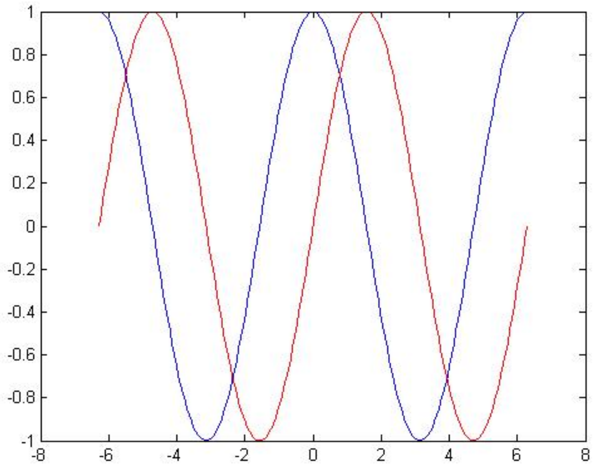
Todimensjonale plot

Kommandoen plot(x,y) lager et todimensjonalt plot, men både x og y må inneholde like mange elementer.

```
>> x=1:10;
>> y=[2 4 7 9 12 10 9 5 2 1];
>> plot(x,y)
```

Vi kan dele opp av x-aksen i like deler med kommandoen **linspace.** Kommandoen hold on brukes for å lage flere kurver på samme plot:

```
x=linspace(-2*pi,2*pi,200);
>> y=cos(1./x);
>> plot(x,y)
>> y=cos(x);
>> plot(x,y)
>> hold on
>> z=sin(x);
>> plot(x,z,'r')
```



Kommandoen plot kan angi linjefarge og spesifisere hvordan linjen skal se ut

plot(x,y,'linjespes','PropertyName',PropertyValue)

Linjespesifikasjon:-- : -. Heltrukken som default

Linjefarger: r g b c m y k w

Markørtyper: + o * . s d p h

Linefarge: color

LineStyle

PropertyName

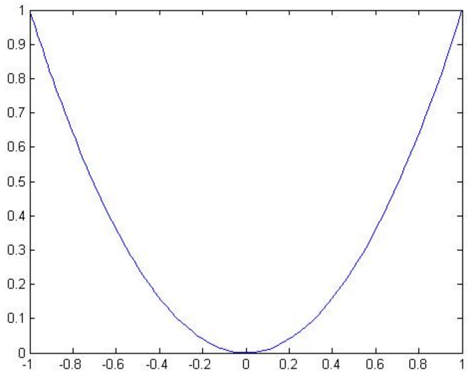
LineWidth MarkerSize MarkerEdge-Color MarkerFace-Color

```
>> plot(x,y,'--r*','linewidth',2,'markersize',14)
```

Kommandoen **fplot** plotter en funksjon f(x):

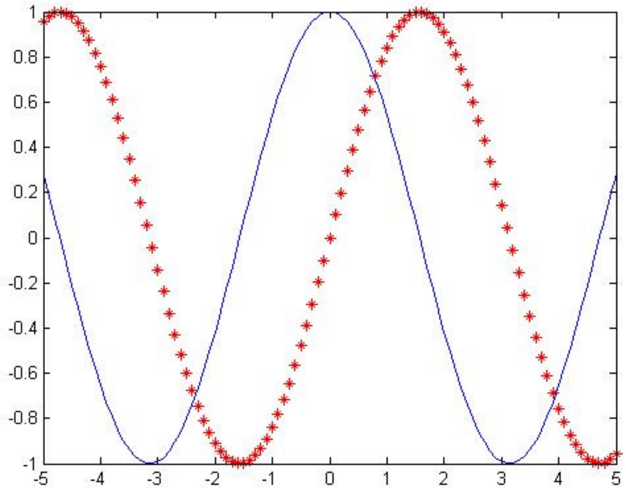
fplot('funksjon',limits,linjespesifikasjon)

```
>> fplot('x^2',[-1 1])
```



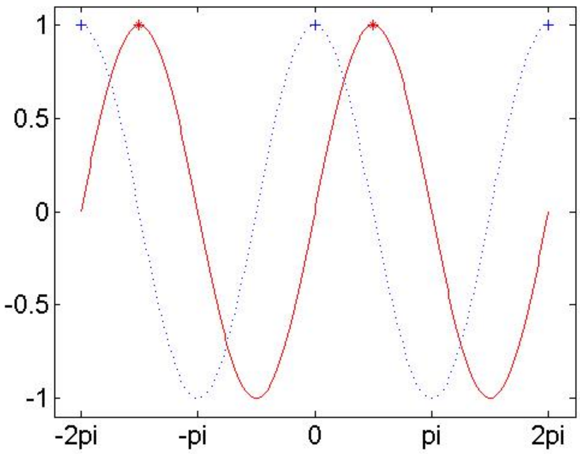
Multiple plot på samme graf:

```
>> x=[-5:0.1:5];
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'*r',x,z,'b')
```

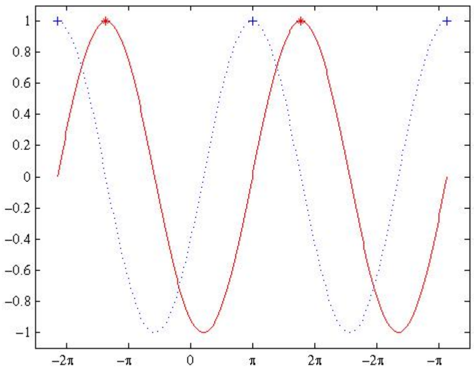


Kommandoen **hold on** og **hold off** hvor aksene beholdes og flere plot legges på etter hvert.
Hvis man skal sette forskjellige verdier av π på aksene:

```
>> x=(-2:0.01:2)*pi;y1=sin(x);y2=cos(x);
>> plot(x,y1,'r-',x,y2,'b:');hold on
>> x1=[-3*pi/2 pi/2];y3=[1 1];plot(x1,y3,'r*')
>> x2=[-2*pi 0 2*pi];y4=[1 1 1];plot(x2,y4,'b+')
>> axis([-7 7 -1.1 1.1])
>> set(gca,'XTick',(-2:2)*pi,'XTickLabel',...
'-2pi|-pi|0|pi|2pi','FontSize',16)
>>
```



Endring er betegnelsene på x-aksen gjøres med kommandoen **set** som kan brukes til å endre diverse egenskaper på figuren og hvis det gjelder aksene må den kombineres med kommandoen **gca** (get current exes)

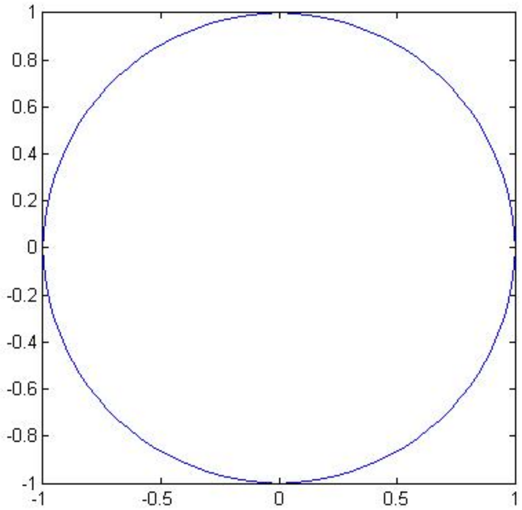


Pi på aksene blir erstattet med symboler vha.

```
>> set(gca,'FontName','Symbol')
>> set(gca,'XTickLabel','-2p|-p|0|p|2p')
```

En sirkel:

```
>> t=0:0.01:1;
>> plot(cos(2*pi*t),sin(2*pi*t))
>> axis square
```



Man kan også lage den med:

>> ezplot('cos(t)','sin(t)',[0 2*pi]);axis square

<

Med kommandoen **line** kan det legges til plot:

line(x,y,'linjetype','–','color','r','marker','*')

Generelt kan man skrive kommandoene **text** (med koordinater hvor teksten skal stå), **xlabel**, **ylabel**, **zlabel**, **legend** og **title**:

Aksetekster med kommandoen **xlabel('Aksetekst')**
ylabel('Aksetekst') **title('navn')**
text(x,y,'tekst')
gtext('tekststreng')
legend('streng1','streng2',,pos)
Pos fra -1 0 1 2 3 4

Teksten kan modifiseres med fonter: \bf \it \rm
\fontname(navn) \fontsize

Greske bokstaver ved å skrive: \name of the letter\alpha \beta osv.
Summetegn \Sigma
Piler kan skrives med \leftarrow eller \uparrow

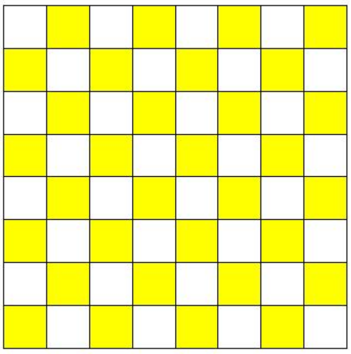
rotation i grader

backgroundColor

>> set(gcf,'Color',[1 0 0])
gir rød bakgrunnsfarge.

Sjakkbrett:
>> red=[1 0 0];yellow=[1 1 0];

```
>> a=[0 1 1 0];b=[0 0 1 1];c=[1 1 1 1];
>> figure,hold on
>> for k=0:1, for j=0:2:6
fill(a'*c+c'*(0:2:6)+k,b'*c+j+k,yellow)
end,end
>> plot(8*a',8*b','k')
>> set(gca,'XtickLabel',[],'YTickLabel',[])
>> set(gcf,'Color',red);axis square
>>
```



Kommandoen **set** sammen med **LineStyle** brukes for å endre utseende av aksene. For øvrig kan grafikk settes inn på figuren med kommandoen **line**, **rectangle**, **fill**, **surface** og **image**.

Akselengden kan endre ved kommandoen **axis**:
axis([xmin,xmax,ymin,ymax])

axis equal gir lik målestokk på begge akser, **axis square** lager kvadratisk vindu, samt **axis tight**.

Rutenett på eller av **grid on** **grid off**

Logaritmisk plot med: **semilogy(x,y)** eller **semilogx(x,y)** eller **loglog(x,y)**.

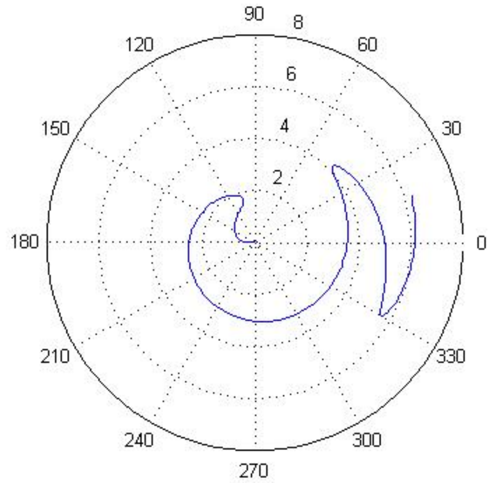
Stolpediagram: **bar(x,y)** **barh(x,y)**

Trappediagram: **stairs(x,y)**
Kakediagram: **pie(x)**
:**stem(x,y)**
Histogram: **hist(y)** **hist(y,nbins)** **hist(y,x)**

Flere plot på same ark: **subplot(x,y,z)**deler vindu i x·y vinduer.
Figuren kommer i et vindu Figur 1. Hvis du ønsker å ha flere figurvinduer tilgjengelig samtidig skriver du:
>>figure og du får tak i Figur 2 ved å skrive >>**figur(2)**.

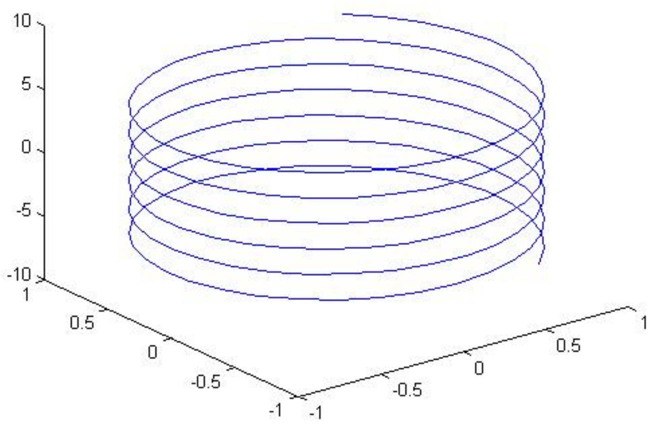
Plot med polarkoordidanter:
polar(theta,radius,'linjespesifikasjon')

```
>> t=linspace(0,2*pi,400);
>> r=3*cos(0.8*t).^2+t;
>> polar(r,t)
```



Bruk av kommandoen **plot3** for et parametrisert tredimensjonalt plot:

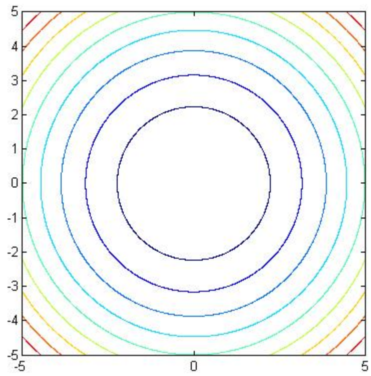
```
>> t=linspace(-2*pi,2*pi,200);
>> r=1;
>> y=sin(pi*t);
>> x=cos(pi*t);
>> plot3(x,y,z)
>>
```



Kontur-plot (contour) ha to variable og er punkter i xy-planet hvor funksjonen har konstant verdi. Konturplot kan lages med kommandoene **meshgrid** og **contour**, hvor meshgrid lager et nettverk av punkter i et rektangel med angitte avstander. I et **tredimensjonalt plot** lager man først et rutenett i xy-planet med kommandoen [x,y]=**meshgrid(a,b)**. Deretter kan det lages plot med **surf(x,y,z)** eller **mesh(x,y,z)** eller alternativt **meshc**.

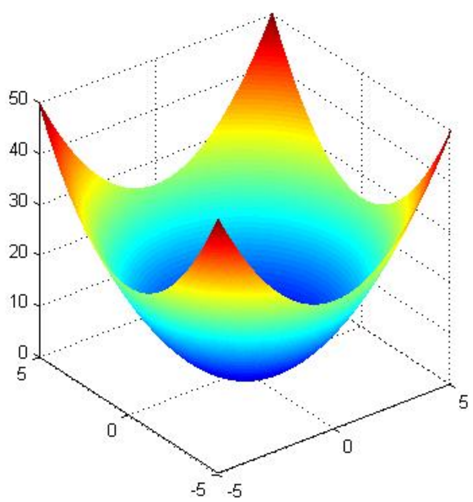
Konturplot av $x^2 + y^2$ blir:

```
>> [x y] = meshgrid(-5:0.01:5,-5:0.01:5);
>> contour(x,y,x.^2+y.^2);axis square
```



Antall konturer kan endres med **contour(x,y,z,n)**. Samme plot med **mesh (x,y,z)** :

```
>> [x y] = meshgrid(-5:0.01:5,-5:0.01:5);
>> mesh(x,y,x.^2+y.^2);axis square
```



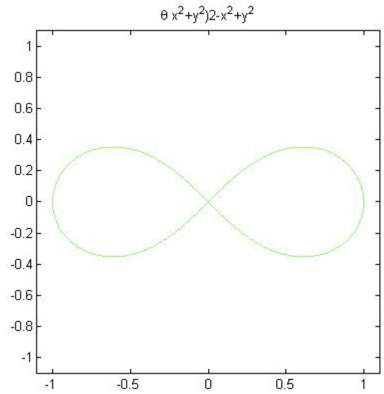
Man kan også plotte sirkler med forskjellige radius :

```
>> contour(x,y,x.^2+y.^2), [1 2 3]
```

Lemniscate (x^2+y^2)2-x^2+y^2) :

```
>> contour(x,y,(x.^2+y.^2).^2-x.^2+y.^2,[0 0])
>> axis square
>> title('\theta x^2+y^2)2-x^2+y^2')
```

Greske bokstaver uttrykkes med en bakoverslash : \alpha

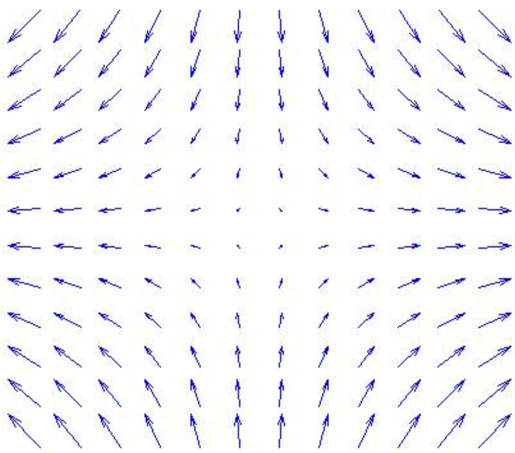


Man kan også lage kontour-plot med kommandoen **ezcontour**.

Felt-plot

Man kan lage vektorfeltplot med kommandoen **quiver**.

```
>> [x y] = meshgrid(-1.1:0.2:1.1,-1.1:0.2:1.1);
>> quiver(x,-y);axis equal;axis off
```



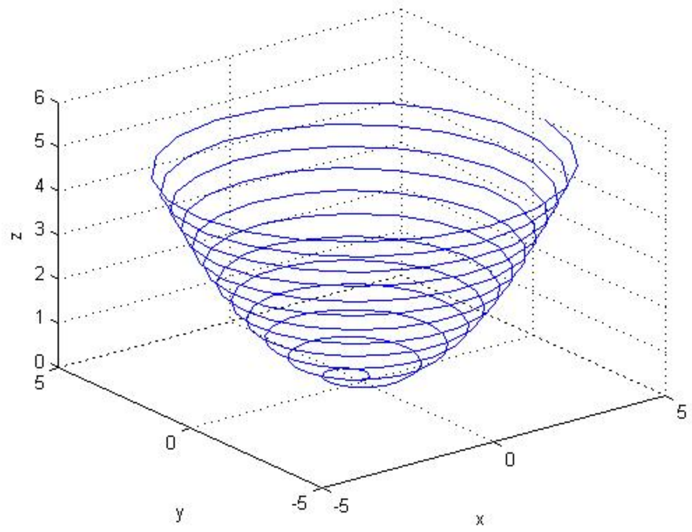
3-dimensjonale plot

3-dimensjonale plot lages med kommandoen plot3
plot3(x,y,z,’linespesifik’,’egenskapsnavn’,egenskapsverdi

Eksempel
Hvis koordinatene x, y og z er gitt ved følgende:
x=sqrt(tcos(3t))
Y=sqrt(tsin(3t))
Z= 0.2 t
Og la t variere fra 0-8pi

```
>> t=0:0.1:8*pi;
>> x=sqrt(t).*cos(3*t);
>> y=sqrt(t).*sin(3*t);
```

```
>> z=0.2*t;
>> plot3(x,y,z,'b','linewidth',1)
>> grid on
>> xlabel('x');ylabel('y');zlabel('z')
>>
```



Mesh og surface plot brukes til å plotte grafer av typen z=f(x,y). x og y er uavhengige variable og z er avhengig variabel.
Først lager man et nettverk i x-y-planet i området som dekkes av funksjonen. Deretter beregnes z for hvert punkt på griden.
Bruker kommandoen **meshgrid**
[X,Y]=meshgrid(x,y)

```
>> x=0:6;
>> y=0:10;
>> [X,Y]=meshgrid(x,y)

X =

     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6
     0     1     2     3     4     5     6


Y =

     0     0     0     0     0     0     0
     1     1     1     1     1     1     1
     2     2     2     2     2     2     2
     3     3     3     3     3     3     3
     4     4     4     4     4     4     4
     5     5     5     5     5     5     5
     6     6     6     6     6     6     6
```

```
     7     7     7     7     7     7     7
     8     8     8     8     8     8     8
     9     9     9     9     9     9     9
    10    10    10    10    10    10    10
```

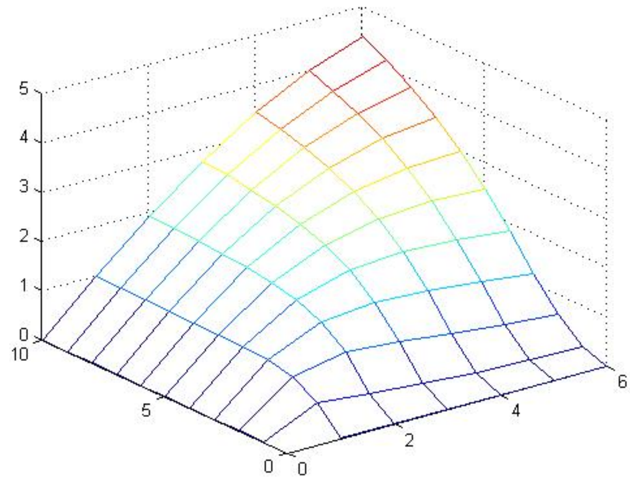
>>Hvis deretter kan Z- beregnes for hvert punkt
for eksempel funksjon $z=xy^2/(x^2+y^2)$

```
>> Z=X.*Y.^2./(X.^2+Y.^2);
```

Et meshplot eller surface plot lages med kommandoene **mesh** og **surf** og disse er de viktigste kommandoene for å plotte flater i 3D.

```
mesh(X,Y,Z)      surf(X,Y,Z)
```

>> mesh(X,Y,Z)

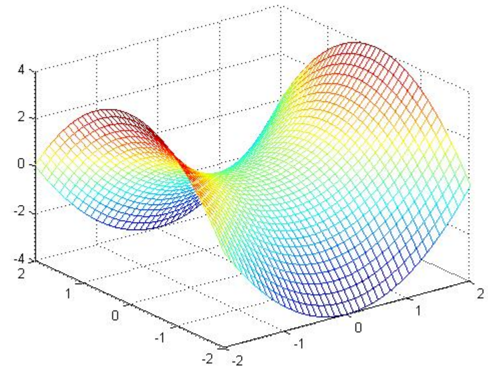


Fargen kan endres ved på bruke Plot Editor i Figure Window eller ved å bruke kommandoen **colormap(C)** hvor C er en vektor som viser intensiteten til fargene R,G,B mellom 0-1

Blå:[0 0 1]
Rød: [1 0 0]
Grønn: [0 1 0]

For eksempel sadelfunksjonen $z = x^2-y^2$:

```
>> [x y]=meshgrid(-2:0.1:2,-2:0.1:2);
>> z=x.^2-y.^2;mesh(x,y,z)
```



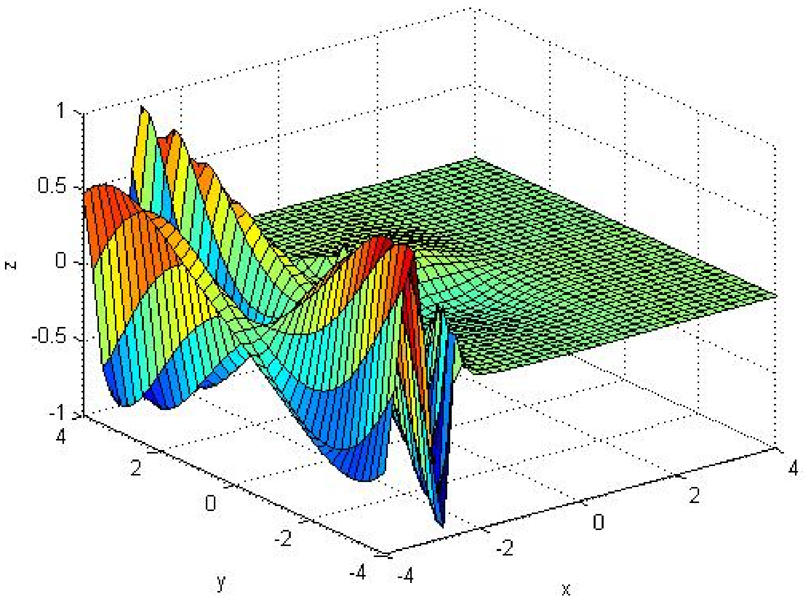
Mesh og grid kan også brukes som mesh(Z) eller surf(Z)

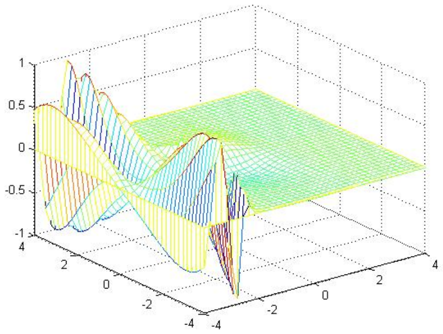
```
>> x=-4:0.2:4;
>> y=-4:0.2:4;
>> [X,Y]=meshgrid(x,y);
>> Z=2.^(-1.8*sqrt(X.^3+Y.^2)).*sin(0.4*Y).*cos(X);
>> surf(X,Y,Z)
>> xlabel('x'),ylabel('y'),zlabel('z')
```

Et alternativ til denne kommandoen er **ezmesh** og **ezsurf**:

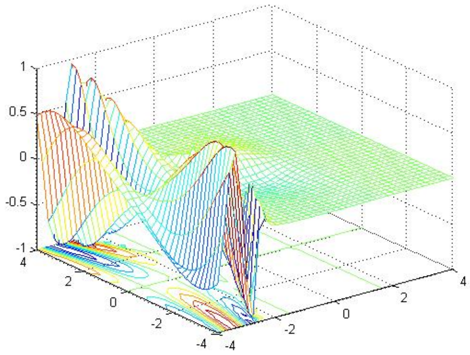
```
>> ezmesh('x.^2-y.^2',[-2,2,-2,2])
```

som gir nøyaktig samme resultat som det over.

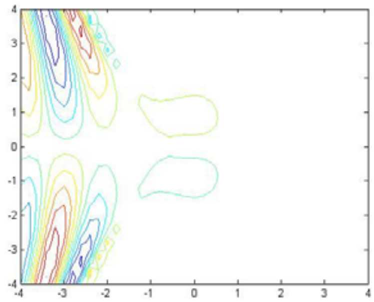




>> meshz(X,Y,Z)

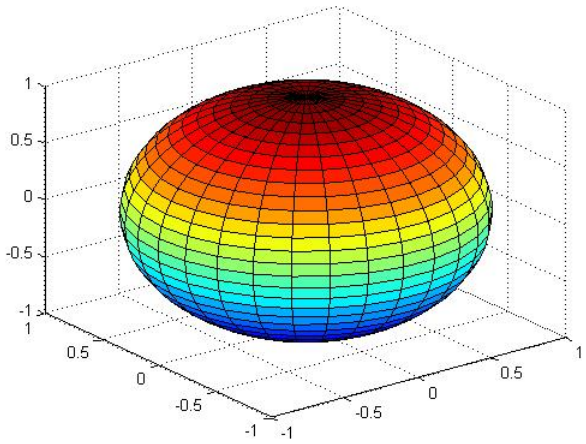


Eksempler på andre typer plot:
>> meshc(X,Y,Z)
>> surfc(X,Y,Z)
>> surfl(X,Y,Z)
>> waterfall(X,Y,Z)
>> contour3(X,Y,Z,12)
>> contour3(X,Y,Z,n)



>> contour(X,Y,Z,12)

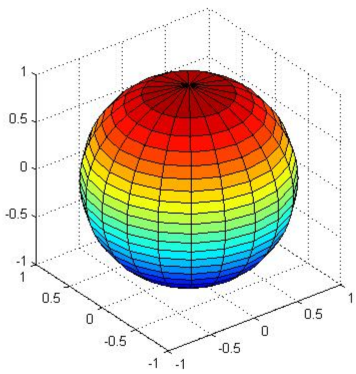
Plotting av kule




```
>> [X,Y,Z]=sphere(30);
>> surf(X,Y,Z)
```

Hvis man ønsker å plotte en funksjon som ikke er av typen *z(f(x,y)* f.eks. $x^2 + y^2 + z^2 = 1$ lønner det seg å lage et kule eller sylinderkoordinatsystem. Hvis ***r*** er avstanden til z-aksen så blir ligningen til kula $r^2 + z^2 = 1$ eller r=sqrt(1-x^2), x=sqrt(1-z^2cosα) og y=sqrt(1-z^2sinα)

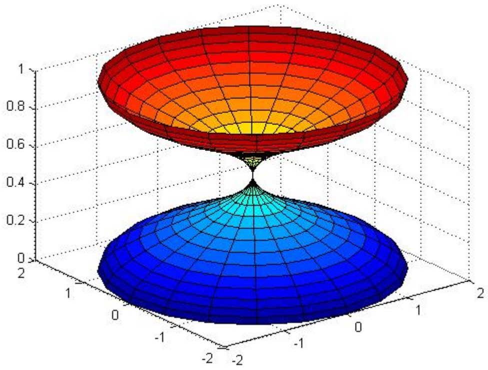
```
>> [z,alpha]=meshgrid(-1:0.1:1,(0:0.1:2)*pi);
>> x=sqrt(1-z.^2).*cos(alpha);
>> y=sqrt(1-z.^2).*sin(alpha);
>> surf(x,y,z);axis square
```



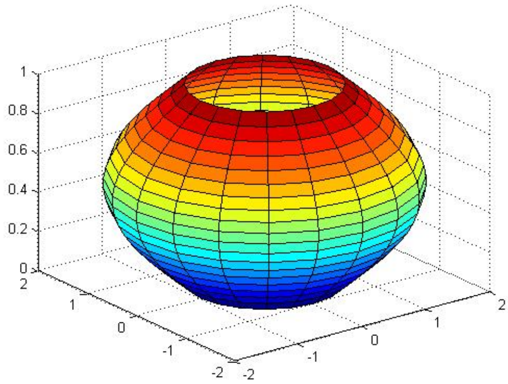
Man kan bruke parameterfunksjonene **ezmesh** og **ezsurf** som gir samme resultat.

Plotting av sylinder :

```
>> t=linspace(0,2*pi,30);
>> r=1+cos(t);
>> surf(X,Y,Z)
>> [X,Y,Z]=cylinder(r);
>> surf(X,Y,Z)
>>
```

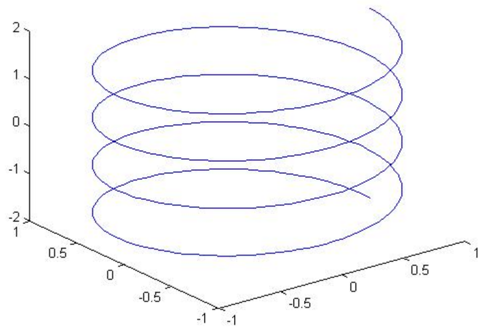



```
>> t=linspace(0,pi,20);
>> r=1+sin(t);
>> [X,Y,Z]=cylinder(r);
>> surf(X,Y,Z)
>>
```

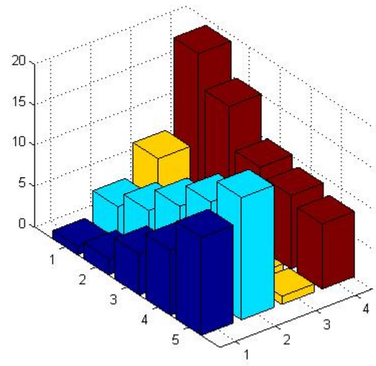


Spiral for x=cos2πz og y=sin2πz:

```
>> t=-2:0.01:2;
>> plot3(cos(2*pi*t),sin(2*pi*t),t)
```

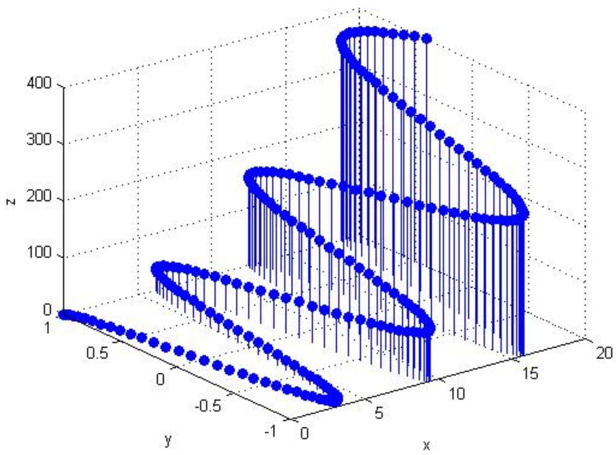


3D-stolpediagram




```
>> Y=[1 4 8 19;2 6 4 15; 5 9 3 10;8 12 1 9;12 15 1 8];
>> bar3(Y)
>>
```

3D-stemplot




```
>> t=0:0.1:20;
>> x=t;
>> y=cos(t);
>> z=t.^2;
>> stem3(x,y,z,'fill')
>> xlabel('x'),ylabel('y'),zlabel('z')
```

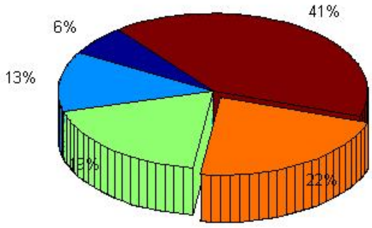
3D-Scatterplot

```
scatter3(X,Y,Z)
```

3D-kakediagram

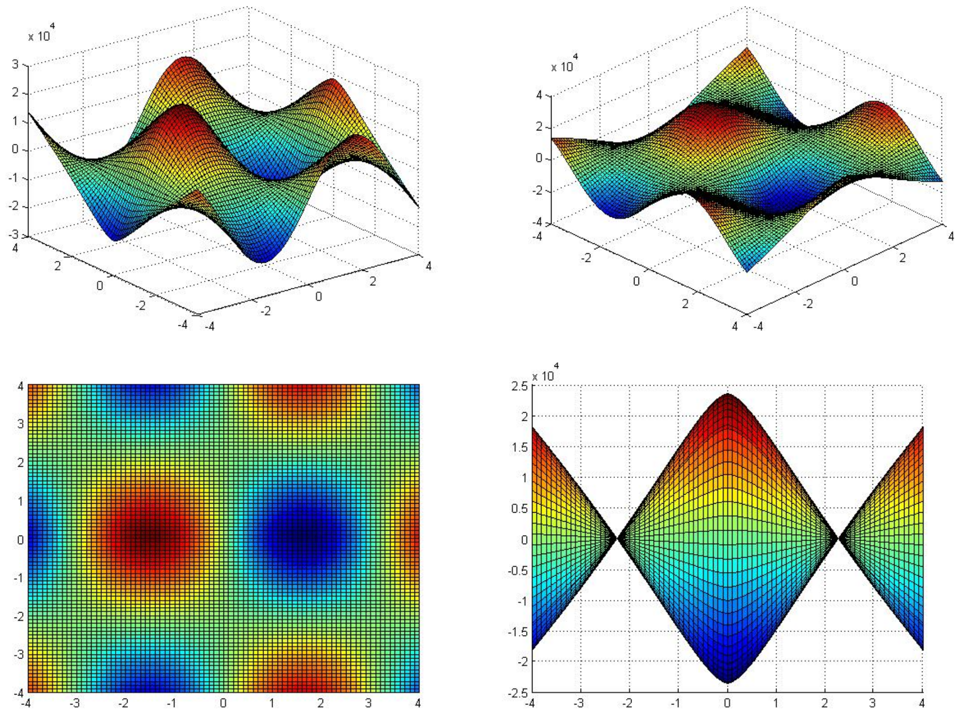
```
>> A= [6 12 18 21 39];
>> explode=[0 0 0 1 0];
>> pie3(A,explode)
```

explode er en vektor som består av 0 og 1 og viser hvilken sektor som skal fjernes fra sentrum av kakediagrammet.



Kommandoen **view (az,el)**eller **view([az,el])** styrer retningen hvorfra plottet observeres angitt ved azimutvinkel i xy-planet og høydevinkel i forhold til xy-planet; az er azimutvinkel i grader og el (elevation) er høydvinkel i grader.

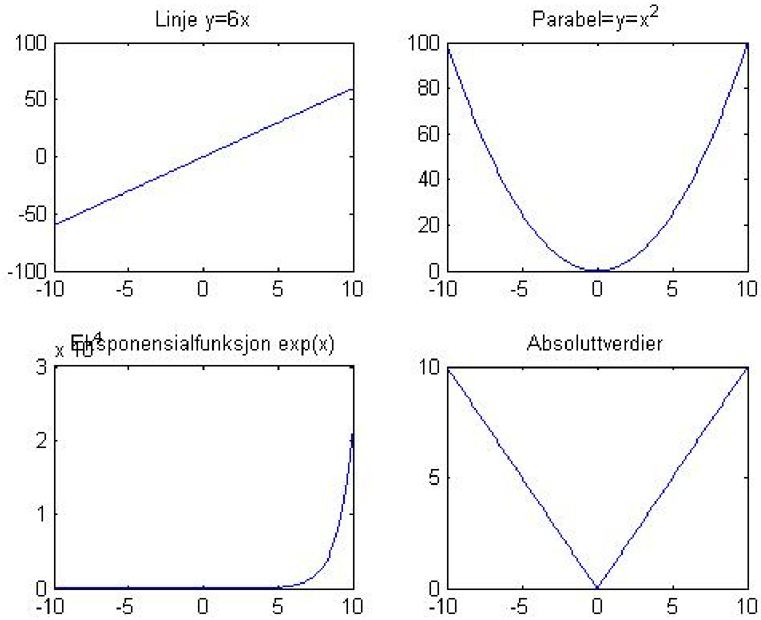
```
>> x=-4:0.1:4;
>> y=-4:0.1:4;
>> [X,Y]=meshgrid(x,y);
>> Z=1.3^(-1*sqrt(X.^2+Y.^2))*cos(X)*sin(X);
>> surf(X,Y,Z)
>> view(45,45)
>> view(0,90)
>> view(90,0)
```



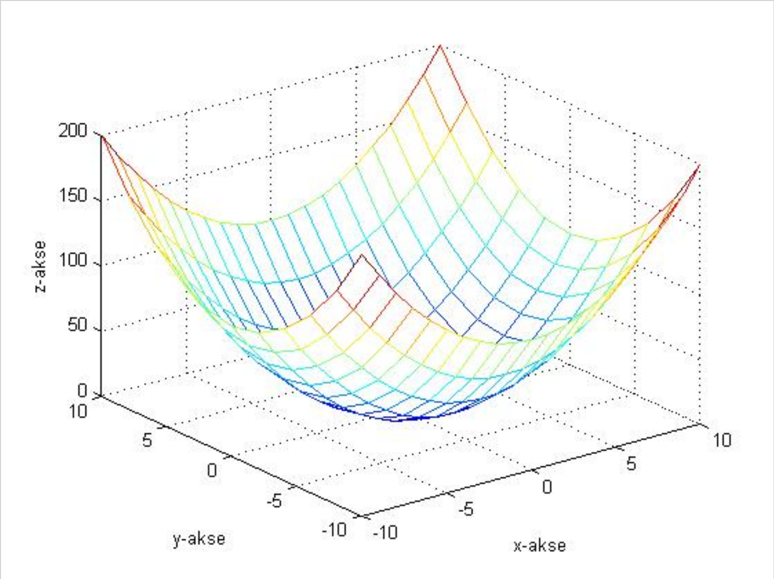
Figurene sett i forskjellig vinkel. Sett ovenfra: az=0 og el=90.

```
>> subplot
```

```
>> x=[-10:0.1:10];
>> %linje y=6x
>> linje=6.*x;
>> %parabel y=x^2
>> parabel=x.^2;
>> %eksponensialfunksjon ex
>> ekspo=exp(x);
>> %Absoluttverdier
>> absol=abs(x);
>> subplot(2,2,1);plot(x,linje);
>> title('Linje y=6x');
>> subplot(2,2,2);plot(x,parabel);
>> title('Parabel=y=x^2');
>> subplot(2,2,3);plot(x,ekspo);
>> title('Eksponensialfunksjon exp(x)')
>> subplot(2,2,4);plot(x,absol);
>> title('Absoluttverdier')
>>
```



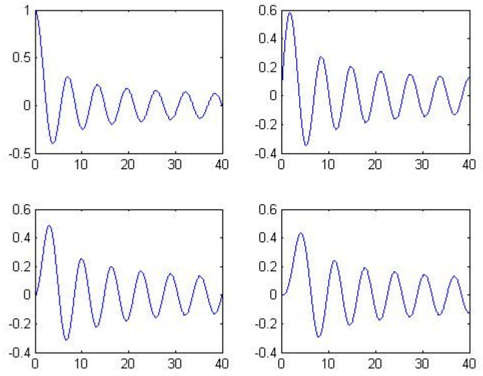

```
>> x= [-10 : 1 : 10];
>> y= [-10 : 2 : 10];
>> [X, Y] = meshgrid(x,y);
>> Z = X.^2 + Y.^2;
>> mesh(X,Y,Z); xlabel('x-akse'); ...
ylabel('y-akse');zlabel('z-akse');
```



Når man lager et plot viser dette seg i et vindu merket Figure 1. Etterfølgende plot endrer eller erstatter det eksisterende plot. Kommandoen hold on gir tilføyelser til grafen. Ved å skrive **figure** åpnes en ny Figure 2, alternativt kan man gå til File og definere en ny figur. Hver figur har en verktøylinje som kan brukes til å editere figuren. Verktøyene kan åpnes med kommandoen **plottools**.

Kommandoen **subplot** deler figurvindu inn i mindre plot. De to første tallene angir dimensjonen på arealet til subplottet og det siste tallet angir hvor mange plot.
For eksempel Besselfunksjonen hvor n=0-4

```
>> x=0:0.01:40;
>> for n=1:4
subplot(2,2,n)
plot(x,besselj(n-1,x))
end
>>
```



Bilder animasjoner og lyd

Matlab kan behandle bilder animasjoner og lyd.
Kommandoen **imread** leser bilde fra forskjellige bildeformater. Hvis man ønsker å lagre et bilde bild.png i formen **rgbpic** blir kommandoen
rgbpic=imread('bilde.png')
Kommandoen image vilder bilde i figurvindu:

```
>>image(rgbpic)
>>axis equal tight
```

Animasjoner med kommandoen comet som lager et parametrisk plot av en kurve, bortsett fra at kurven vises over tid.

```
>> t=(0:0.01:2)*pi;
>> figure,axis equal,axis([-1 1 -1 1]),hold on
>> comet(cos(t),sin(t))
```

Mer komplekse animasjoner med kommandoene **getframe** og **movieview**.

```
>> x=0:0.1:1;
>> for n=0:50
plot(x,sin(n*pi/5)*sin(pi*x)),axis([0, 1, -2, 2])
M(n+1)=getframe;
End
>> movieview(M)
```

Hvis man har en film kan man bruke **movie2avi** for å lagre den som avi.-fil
For eksempel >>moview2avi(M,'streng.avi')

Lyd

Man kan bruke kommandoen sound for å lage lyd hvor kommandoen lager en vektor og ser på den som en bølgeform og spiller den. Sinusvektor tilsvarer en ren tone og frekvensen på sinussignalet bestemmer pitch. Følgende er fra Beethovens femte symfoni:

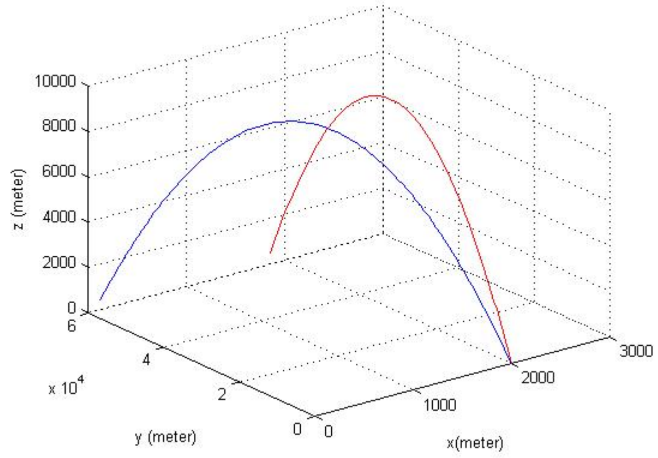
```
>> x=(0:0.1:250)*pi;y=zeros(1,200);z=(0:0.1:1000)*pi;
```

```
>> sound([sin(x),y,sin(x),y,sin(x),y,sin(z*4/5),y,...
sin(8/9*x),y,sin(8/9*x),y,sin(8/9*x),y,sin(z*3/4)]);
```

Matlab kan bare lese og skrive wav og au-format. Kommandoen **wavread** og **wavwrite** leser og skriver disse formatene.

Eksempel:
En jaktpatron har utgangshastighet v0=850 m/s skutt rett mot nord uten vind og med sidevind fra vest 20 m/s skytes ut i vinkel α 30o i forhold til planet. Vi kan la x peke mot øst og y peke mot nord. La utskytingsstedet ha kooridantene x0,y0,z0= (2000,0,0)
Initialhastigheten v0 kan deles i en horisontal y-komponenet og en vertikal z-komponent:
V0y)v0cos(α) og v0z=v0sin(α)
Posisjon z er gitt ved vz=v0z – gt
Og z=z0+v0zt-1/2gt2
Tiden det tar for å nå høyeste punkt i kulebanen dvs. vz=0 er tmax=v0z/g
Total tid ttot=2tmax.
I horisontalretning er hastigheten konstant og posisjonen til prosjektilet er gitt ved
X=x0+vxt og y=y0+v0yt

```
>> v0=850;g=9.81;alfa=30;
>> x0=2000;vx=-20
>> v0z=v0*sin(alfa*pi/180);
>> v0y=v0*cos(alfa*pi/180);
>> t=2*v0z/g;
>> tplot=linspace(0,t,100);
%Lager en vektor med 100 elementer
>> z=v0z*tplot-0.5*g*tplot.^2;
>> y=v0y*tplot;
>> x=x0+vx*tplot;
%Beregner x, y og z koordinater til enhver tid
>> xnowwind(1:length(y))=x0;
%Konstant x-koordinat hvis ikke vind
>> plot3(x,y,z,'b',xnowwind,y,z,'r')
>> grid on
>> axis([0 3000 0 60000 0 10000])
>> xlabel('x(meter)');ylabel('y (meter)');zlabel('z
(meter)')
```



Funksjoner og funksjonsfiler

Lages i Editor-vindu og første linje på definere funksjonen. Kommandoen function sier at det er en funksjon, definerer funksjonsnavn og antall og orden på input og output til funksjonen:
function[output]=funksjonsnavn(input)

Eksempel eksponensiell vekst:
$N(t)=N_o e^{kt}$
K=1/t1 ln Nt1/N0

```
function Nt=expVekst(N0,Nt1,t1,t)
%ekspVekst begregner eksponensiell vekst og nedbrytning
%inputvariable er:
%N0: er mengden ved tid=0
%Nt1: er mengde ved tid t1
%t1: tid t1
%outputvariabel er:
%Nt er mengde ved tid t
k=log(Nt1/N0)/t1;
Nt=N0*exp(k*t);
>>expVekst(67,79,6,20);
```

Programmering i Matlab

Løkker

Løkker starter med **for** og slutter med **end**. Hvis du bruker løkker i en script m-fil med **echo on** så kan du fjerne dette ved å sette **echo off** like før end.

```
for k= f:s:t
>> for k=1:2:10
x=k^2
end
x =
     1
x =
     9
x =
    25
x =
    49
x =
    81
<
```

while-end løkker:

```
>> x=1;
>> while x<=14
x=2*x
end
x =
     2
x =
     4
x =
     8
x =
    16
>>
```

Beregning av de 10 første Fibonacci-tallene:

```
>> f=[1 1];%f(0)=1 og f(1)=1
>> for n=3:10;%starter på det tredje tallet
f(n)=f(n-1)+f(n-2);
end
>> f

f =
     1     1     2     3     5     8    13    21    34    55

>>
```

Logiske operatorer er: **&** (og), **|** (eller) og **~** (ikke)
Symboler: **<** (mindre enn), **>** (større enn),**==** (lik),**<=** (mindre enn eller lik), **>=** (større enn eller lik), samt ~= (ikke lik)
Logiske operatorer:

```
if else then
if elseif else end
if end
AND  OR  NOT
Xor(a,b)
```

Kommandoene **break** og **continue**.

Simulink startes ved å skrive kommandoen **simulink**.

M-filer

M-filer kan brukes til å lagre flere kommandoer i en fil og deretter kjøre dem. Det finnes to typer M-filer:
Script M-filer
Funksjons M-filer
Den første linjen i funksjons-M-filer er en funksjonsdefinisjon.

Det går an å publisere M-filer med kommandoen **publish** som omdanner M-filen til et lesbart dokument.
Hvis du skal lage en interaktiv M-fil bruk kommandoen **pause**.
Lager en M-fil i editor-vindu:

```
%M-script-fil
%Beregner sin(x)/x for x= 0.1, 0.01 og 0.001
clear all %fjerner alle gamle variabelnavn
format long  %lager 15 desimaler
x=[0.1, 0.01, 0.001] %definerer x-verdier
y=sin(x)./x %definerer y-verdier
%Disse verdiene viser
x =
   0.10000000000000   0.01000000000000   0.00100000000000
y =
   0.99833416646828   0.99998333341667   0.99999983333334
```

```
>> publish test3
ans =
C:\MATLAB7\work\html\test3.html
>>
```

```
    %M-script-fil
    %Beregner sin(x)/x for x= 0.1, 0.01 og 0.001
    clear all %fjerner alle gamle variabelnavn
    format long % lager 15 desimaler
    x=[0.1, 0.01, 0.001] %definerer x-verdier
    y=sin(x)./x %definerer y-verdier
    %Disse verdiene viser
x =
   0.10000000000000   0.01000000000000   0.00100000000000
y =
   0.99833416646828   0.99998333341667   0.99999983333334
```

Published with MATLAB® 7.0

Hvis du skriver lange kommandolinjer og linjen blir for lang kan du avslutte linjen med tre prikker … og deretter trykke enter og så kan du fortsette på neste linje.

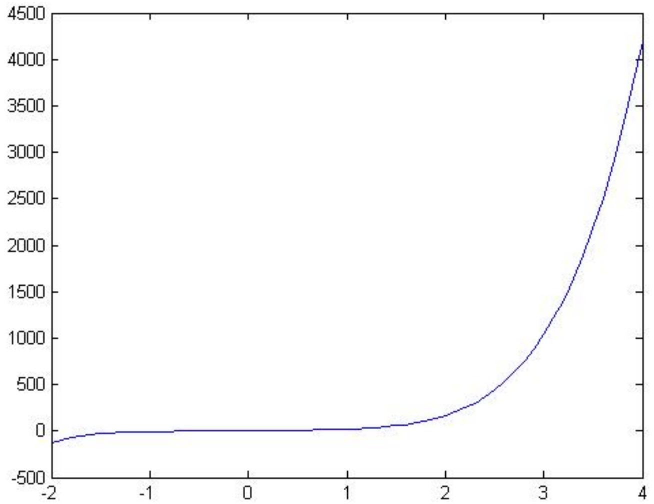
Polynomer og kurveglatting

$f(x)=4x^5 + 4x^2 + 9x + 3$ c=[4 4 9 3]
Finn f(8):
Kommandoen **polyval(p,x)** hvor p er en vektor med koeffisientene til polynomet og x er et tall eller variabel med en bestemt verdi, dvs, verdien til polynomet ved punkt x

```
>> c=[4 4 0 0 9 3];
>> polyval(c,8)
ans =
        131403
```

Plotte polynomet:

```
>> x=-2:0.1:4;
>> y=polyval(c,x);
>> plot(x,y)
```



Kommandoen **roots** bestemmer røttene til et polynom, spesielt viktig for kvadratiske ligninger.

```
>> r=roots(c)
r =
   0.9316 + 1.0150i
   0.9316 - 1.0150i
  -0.7317 + 0.6730i
  -0.7317 - 0.6730i
  -0.3998
>>
```

Hvis røttene til et polynom er kjent kan man bruke kommandoen **poly** for å finne koeffisientene til polynomet.

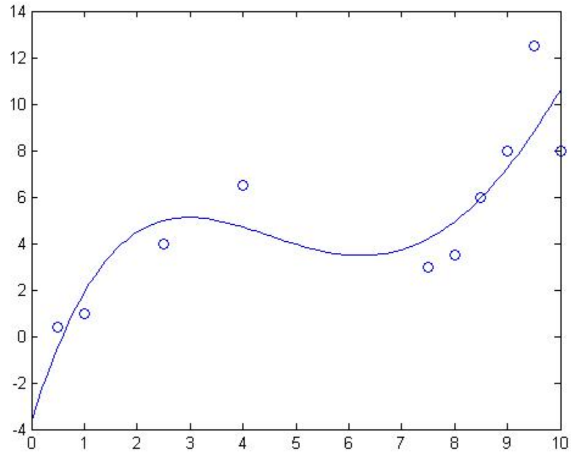
Polynomer kan multipliseres med hverandre med kommandoen **conv**
D=conv(a,b)
Eller divideres på hverandre med **deconv**

Derivering av polynomer med **polyder**
K=polyder(p) deriverte av et polynom. P er koffisientvektor
K=polyder(a,b) deriverte av produktet av to polynomer
K=polyder(u,v)

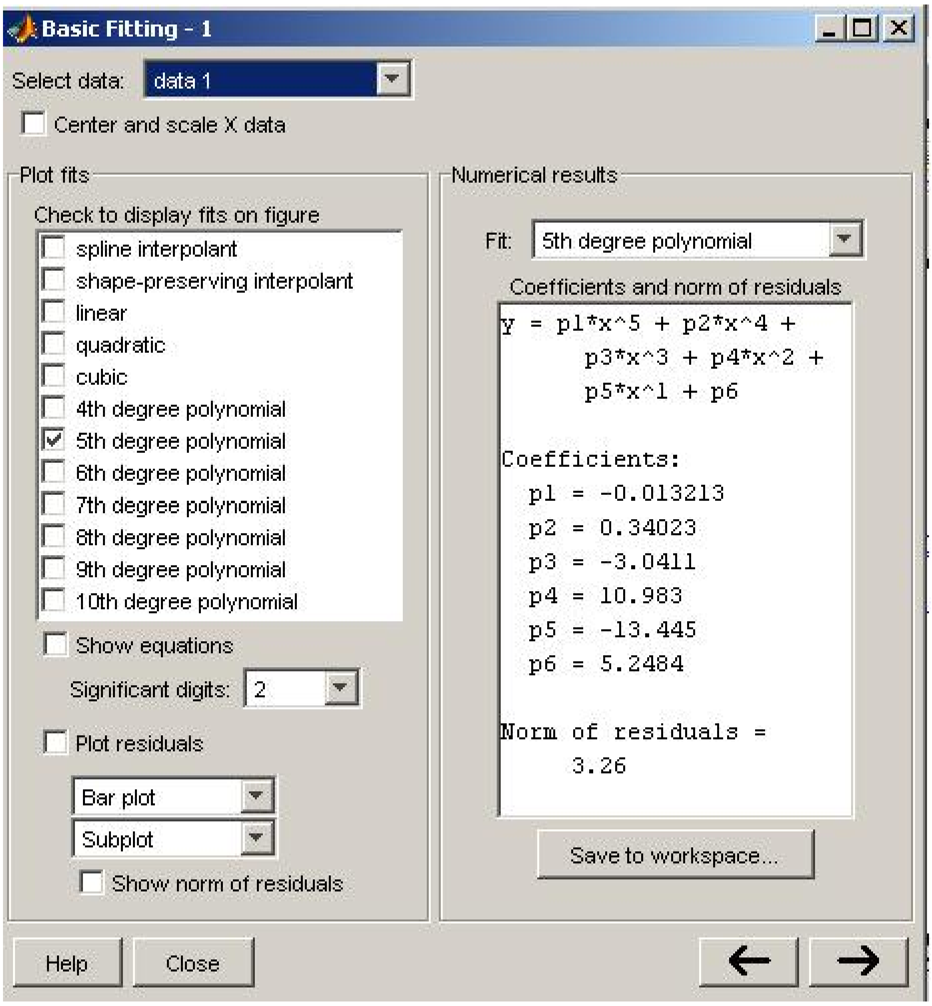
Kurveglatting

Kurveglatting er en form for regresjonsanalyse.
Kurveglatting av polynomer via **polyfit**
Som baserer seg på minste kvadraters metode

```
>> x=[0.5 1 2.5 4  7.5  8  8.5  9  9.5 10];
>> y=[0.4 1 4 6.5 3 3.5 6 8 12.5 8];
>> p=polyfit(x,y,4)
p =
   -0.0065    0.2141   -2.0964    7.4650   -3.6686
>> xf=0:0.1:10;
>> yf=polyval(p,xf);
>> plot(x,y,'o',xf,yf,'b')
```



$-0.0065x^4 + 0.2141x^3 - 2.0964x^2 + 7.465x - 3.6686$



Kurveglatting for andre funksjoner enn polynomer:
Y=bx^{m} –potens
Y=be^{mx} eller y=b10^{mx} eksponensialfunksjon
Y=mln(x)+b eller y=mlog(x) + b Logaritmefunksjon
Y=1/(mx + b) resiprok funksjon.

Disse omskrives slik at de kan brukes med polyfit i en type y=mx + b

Ln(y)= mln(x) + lnb potensfunksjon
Ln(y) = mx + ln(b) eller log(y) = mx + log(b)
logfunksjon
1/y= mx + b resiprok

Potens : p=polyfit(log(x),log(y),1)
Eksponensial : p=polyfit(x,log(y),1 eller
P=polyfit(x,log10(y),1)

Logfunksjon : p=polyfit(log(x),y,1)
p=polyfit(log10(x),y,1)

Resiprok : p=polyfit(x,1./y,1)

Man kan også velge fra Tools i plottemenyen.

Interpolering

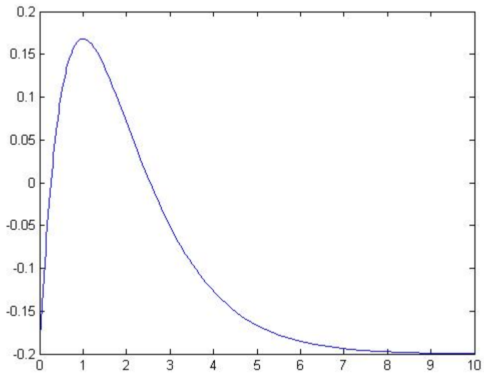
Interpolering er bestemmelse av verdier mellom datapunkter, som kan bestemmes via polynomer eller via Fourier-transformasjoner.
Endimensjonal interpolering via spline. I denne trekkes det en linje mellom to punkter som ligger ved siden av hverandre og man bruker kvadratisk eller kubisk spline.
Endimensjonal interpolering med funksjonen **interpl**
Interlpolert verdi yi: yi=interpl(x,y,xi,’method’)
Metodene som kan brukes er nearest linear spline pchip

Numerisk analyse og funksjoner

Skjæring med x-aksen

Har man en ligning med en variabel kan den skrives f(x)=0 hvor løsningen er når den krysser x-aksen. Virker bare for funksjoner som krysser x-aksen. Hvis det er vanskelig å finne x kan man gjøre iterasjoner slik at man nærmer seg. Prosessen kan stoppes når verdien til x mellom to iterasjoner er mindre enn en fastsatt verdi.0-verdien til en funksjon beregnes med kommandoen **fzero**
x=fzero(’funksjon’,x0)
Hvor x er løsningen, funksjon er funksjonen som skal løses og den må skrives på formen f(x)=0 og x0 er en verdi for x nær der funksjonen krysser aksen.

Bestem løsningen av ligningen $x*e^{-2*x}= 2$
$f(x)= x*e^{-*x}- 0.2$
>> fplot('x*exp(-2*x)-2',[0 5])



Første skjæringspunkt I nærheten av 1 og den andre i nærheten av 3:
>> fplot('x*exp(-x)-0.2',[0 10])
>> x1=fzero('x*exp(-x)-0.2',1)

```
x1 =

    0.2592

>> x2=fzero('x*exp(-x)-0.2',3)

X2 =

    2.5426

>>
```

Finne maksimum og minimum til en funksjon

Bruker kommandoen **fminbnd**

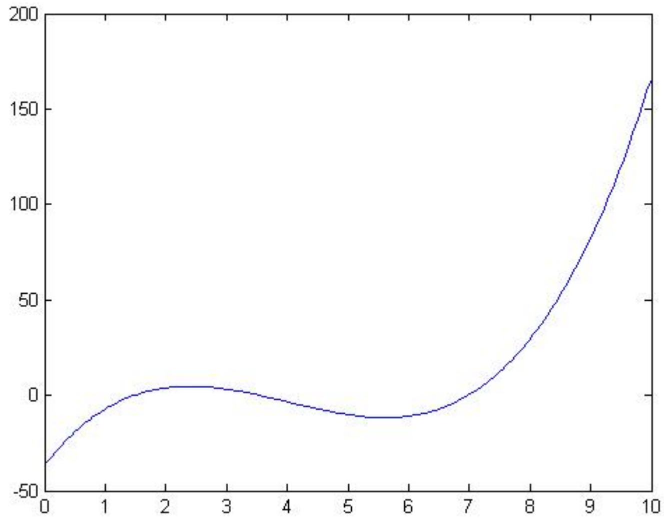
x=fminbnd('funksjon',x1,x2)

X1 og x2 er intervallet

Man kan også finne funksjonsverdien ved minimumsverdien ved

[x favl]=fminbnd('funksjon',x1,x2)

```
>> fplot('x^3-12*x^2+40.25*x-36.5',[0 10])
```




```
>> [x fval]=fminbnd('x^3-12*x^2+40.25*x-36.5',0,10)
x =
     0
fval =
  -36.5000
>> [x fval]=fminbnd('x^3-12*x^2+40.25*x-36.5',3,8)
x =
   5.6073
fval =
  -11.8043
<
```

Numerisk integrasjon

Integrasjon vil si å finne arealet under en funksjon, mellom funksjonen og x-aksen og mellom to grenser a og b, og brukes til å beregne arbeid fra kraft, hastighet ut fra aksellerasjon eller å beregne areal eller volum. Kompliserte funksjoner er nesten umulig å integrere analytisk. Følgende kommandoer kan brukes til integrering **quad** og **quadl** som brukes når f(x) er en funksjon og **trapz** når f(x)er gitt ved datapunkter. Hvis q er verdien til integralet:

q=quad('funksjon',a,b)

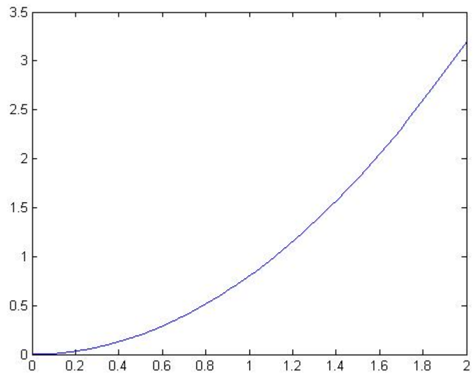
Funksjonen skrives på samme måte som fzero. Beregningen av integralet har en absolutt feil mindre enn $1 \cdot 10^{-6}$, men denne kan endres med tol.

q=quad('funksjon',a,b,tol)

q=trapz(x,y)

hvor xy og y er vektorer med x og y koordinater til punktene, og vektorene må ha samme lengde.

Integralet til $y=0.8 \cdot x^2$ mellom 0 og 1




```
>> q=quad('0.8.*x.^2',0,1)
q =
   0.2667
```

En annen måte er å definere en funksjon:

Ordinære differensialligninger (ODE)

Differensialligninger er viktige i alle naturvitenskapene. Det er få differensialligninger som kan løses analyttisk. Det finnes et stort bibliotek med verktøy som kan brukes til å løse differensialligninger numerisk. Vi starter med løsning av første ordens differensialligninger. En ODE er en ligning som inneholder en uavhengig variabel (x), avhengig variabel (y) og den deriverte av avhengig variabel:

dy/dx= f(x,y).

Vi kan starte med et eksempel hvor t er uavhengig variabel:
dy/dt=f(t,y) for t0≤t≤t1 med y=y0 ved t=t0
Denne uttrykker den deriverte av y med hensyn til t .

```
>> help ode45
```
Kommandoen **ode45** kan brukes til å løse differensialligninger og den kan være av typen:
Ode45(f, [0 2],1) eller ode45(@func,[0 2],1)

Løs ligningen dy/dt= $(t^2-2y)/t$ for 1≤t≤4 hvor y=3 5 ved t=1
Først lager man en funksjonsfil:
```
<<function dydt=ODEexp1(t,y)
>>dydy=(t^2-2*y)/t;
>>[t,y]=solver_name(‘ODEfun’,tspan,y0=
>>[t,y]=ode45(‘ODEexp1’,[1 :0.5 :3],5)
>>plot(t,y)
```

Matematikk og symbolregning

Regning med symbolske operatorer, men har samme form som de numeriske operasjonene. For å sjekke at denne er installert skriv ver eller **help symbolic**

Et symbolsk objekt kan være en variabel eller et tall.
objektnavn=sym(’streng’)

```
>> x=sym('x');
>> %lager et symbolsk objekt x
>> a=sym(12);
>> %a er et symbolsk objekt av 12
>> syms x y z
>> %lager variable x y og z men viser dem ikke før du
skriver dem
>> syms a b c x y
>> f=3*x^3+4*b*x^2+c*x

f =

3*x^3+4*b*x^2+c*x
```

Kommandoen **findsym(S)** eller **findsym(S,n)** brukes for å vise alle symbolvariable.

Kommandoen **collect(S)** eller **collect(S,variabelnavn)** samler alle variable med samme potens, expand og factor

```
> format compact
>> syms x
>> S=(x^3+3*x^2+exp(x)-sin(x))*(2*x+5)
```

```
S =
(x^3+3*x^2+exp(x)-sin(x))*(2*x+5)
>> A=collect(S)
A =
2*x^4+11*x^3+15*x^2+(2*exp(x)-2*sin(x))*x+5*exp(x)-5*sin(x)
>> syms x y
>> S=(x^3+3*y^2+exp(x)-sin(y))*(2*x+5)
S =
(x^3+3*y^2+exp(x)-sin(y))*(2*x+5)
>> B=collect(S,y)
B =
(6*x+15)*y^2+(x^3+exp(x)-sin(y))*(2*x+5)
>>
```

Kommandoen **expand** ekspanderer og multipliserer ut uttrykk.

```
>> K=(x+7)*(2*x+a)*(x+2)
K =
(x+7)*(2*x+a)*(x+2)
>> L=expand(K)
L =
2*x^3+18*x^2+x^2*a+9*x*a+28*x+14*a

>> expand(cos(x+y))
ans =
cos(x)*cos(y)-sin(x)*sin(y)
>>
```

Kommandoen **factor(S)** endrer et polynom slik at det blir et produkt av polynomer med lavere potens

Kommandoen **simplify** forenkler et uttrykk
Kommandoen **simple** finner det uttrykket med færrest
Kommandoen pretty viser et symboluttrykk i et matematisk form.

Kommandoen **l=solve(ligning)** eller **l=solve(ligning,variabel)** løser ligninger.
Solve kan også brukes til å løse flere ligninger
l=solve(lign1,lign2,lign3)

```
>> solve('2*x^2-3*x+3=0')
ans =
 3/4+1/4*i*15^(1/2)
 3/4-1/4*i*15^(1/2)
>> %Her er svaret komplekse tall
```

Du kan plotte en funksjon:

```
>> ezplot(@(x) 2.*x.^2 + 3.* x+2,[-5,5])
```

Du kan plotte en vektor med numeriske data:

```
>> x=[1 2 3 4 5];y=[2,4,6,8,10];plot(x,y)
```

Skal man lage flere plot på samme figur bruk hold on og hold off.

```
>> ezplot('exp(x)',[0 10])
>> hold on
>> ezplot('x.^2',[0 10])
```

Men man kan også plotte flere funksjoner på samme graf ved :

```
>> x=0:0.1:10;plot(x,exp(-x),x,sin(x))
```

Derivering utføres med kommandoen **diff(S)** eller **diff(S,variabel)** for eksempel **diff(S,2)** beregner den andrederiverte

```
>> syms x
>> S=sin(x)
S =
sin(x)
>> diff(S)
ans =
cos(x)
```

Eller alternativer:

```
>> syms x,diff(x^3)
ans =
3*x^2
>> f=@(x) x^3;diff(f(x))
ans =
3*x^2
>>
```

Integrasjon utføres med kommandoen **int(S)** eller int(S,variabel).

Beregner et ubestemt integral:

```
>> int('x^2','x')
ans =
1/3*x^3
```

Selvsagt kan ikke alle funksjoner integreres symbolsk og da må det gjøres numerisk med kommandoene quad og quadl.

Beregning av et dobbeltintegral fra 0-pi og 0-sinx for $(x^2+y^2)dxdy$

```
>> syms x y;int(int(x^2 + y^2,y,0,sin(x)),0,pi)
ans =
pi^2-32/9
>>
```

Det er også en kommando dblquad som kan brukes.

Grensene for et integral kan settes med kommandoen **limit**.

```
>> syms x;limit(sin(x)/x,x,0)
ans =
1
```

Man kan også beregne en-sidige grenser med 'right' og 'left'

Summer og produkter

Summer og produkter kan lett beregnes med kommandoene **sum** og **prod**

```
>> x=1:100;
>> sum(x)
ans =
        5050
```

Det var en fortelling om Gauss som ikke ville sitte rolig på pulten og fikk i oppgave og legge sammen alle tallene fra 1-100, 99+1,98+2 osv., svaret kom umiddelbart.

```
>> prod(x)
ans =
    9.332621544394410e+157
<
```

Man kan finne endelige og uendelige symbolsummer med kommandoen symsum som bruker variabelen k som default.

$$\sum_{k=1}^{n}\left(\frac{1}{k}-\frac{1}{1+k}\right)$$

```
>> syms k n;symsum(1/k - 1/(k+1),1,n)
ans =
-1/(n+1)+1
>>
```

$$\sum_{n=1}^{\infty}\frac{1}{n^2}$$

```
>> symsum(1/n^2,1,inf)
ans =
1/6*pi^2
```

Eller den uendelige geometriske rekke a^k:

```
>> syms a k;symsum(a^k,0,Inf)
ans =
-1/(a-1)
```

Forutsatt at -1<a<1

Matlabs default variabel for solve er x, og hvis du skal løse for en annen må det angis:

```
>> solve('x+y=3')
ans =
-y+3
>> solve('x+y=3','y')
ans =
-x+3
>>
```

Taylor-rekker

Kommandoen **taylor** lager Taylor-polynom rekkeutvikling av spesifikk grad på et spesifikt punkt. For eksempel Taylorpolynomet av sin(x) i grad 9 ved x=0

```
>> syms x;taylor(sin(x),x,10)
ans =
x-1/6*x^3+1/120*x^5-1/5040*x^7+1/362880*x^9
>>
```

Man kan også finne Taylorpolynom ved et annet utgangspunkt enn origo:

```
>> taylor(exp(x),4,2)
ans =
exp(2)+exp(2)*(x-2)+1/2*exp(2)*(x-2)^2+1/6*exp(2)*(x-2)^3
<
```

Man kan også beregne Taylorutvikling til uendelig:

```
>> taylor(exp(1/x^2),6,Inf)
ans =
1+1/x^2+1/2/x^4
>>
```

Differensialligninger

Differensialligninger (ODE) kan løses med kommandoen **dsolve('ligning')** eller **dsolve('ligning','variabel')**

Vi kan starte med et eksempel hvor t er uavhengig variabel og y er avhengig variabel
dy/dt=f(t,y)

Andre ordens ODE inneholder den andrederiverte av avhengig variabel og den første deriverte og får formen
$d^2y/dt^2=f(t,y,dy/dt)$

Alle bokstaver kan brukes, men ikke D for avhengig variabel, fordi D betyr derivering, for eksempel Dy=dy/dt. Eksempel: dy/dt+5y=30 blir 'Dy+yy=30'.

D2 betyr andrederiverte slik at d2y/dt+4dy/dt+4y=cos(t) blir 'D2y+4*D+4y=cos(t)'

C1, C2 osv brukes som konstanter ved integrasjon

For eksempel kan man løse ligningen dy/dt=4t+2y

```
>> dsolve('Dy=4*t+2*y')
ans =
-2*t-1+exp(2*t)*C1
>>
```

En generell løsning av $d^2x/dt^2+2dx/dt+x=0$ blir

```
>> dsolve('D2x+2*Dx+x=0')
ans =
C1*exp(-t)+C2*exp(-t)*t
>>
```

Vi kan for eksempel løse ligningen ds/dt=ax^2

```
>> dsolve('Ds=a*x^2')
ans =
x^2*a*t+C1
>>
```

Hvor løsningen s=ax^2t + C1 er løsningen.

Hvis vi definerer *x* som uavhengig variabel i den samme ligningen

```
>> dsolve('Ds=a*x^2','x')
ans =
1/3*x^3*a+C1
>>
```

Så blir løsningen x=1/3ax^3 + C1

Hvis vi definerer *a* som uavhengig variabel i den samme ligningen blir løsningen:

```
>> dsolve('Ds=a*x^2','a')
ans =
1/2*x^2*a^2+C1
```

Så blir løsningen s=1/2a^2x^2+C1

Det går også an å finne en spesiell løsning for en differensialligning
dsolve('ligning','betingelse','variabel) eller hvis det er flere betingelser
dsolve('ligning','beting1','beting2','var')

Eksempel vi har en første ordens ODE av formen
y'=ay + b dvs.
dy/dt=ay+b
hvor a og b er konstanter hvor en generell løsning er:
y=ce^{at}-b/a
For eksempel dy/dt=3y-30
'Dy-3*y=-30'

```
>> dsolve('Dy-3*y=-30')
ans =
10+exp(3*t)*C1
>>
```

Hvis vi har ODE dy/dt+4y=60 med startverdi y(0)=5

```
>> dsolve('Dy+4*y=60','y(0)=5')
ans =
15-10*exp(-4*t)
>> Som betyr:
```

y=15-$10e^{-4t}$
>

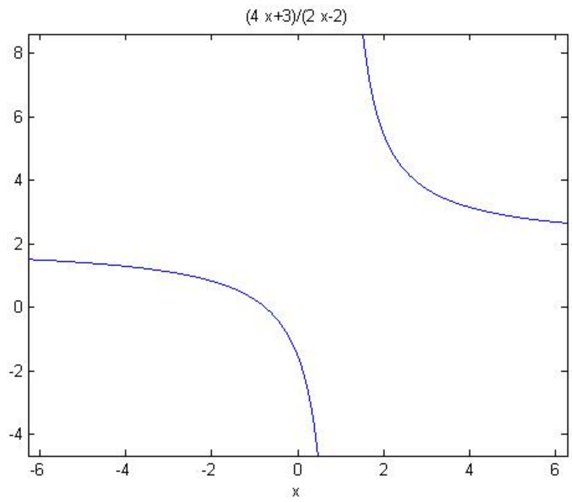
En andre ordens differensiallligning
$d^2y/dt^2 - 2dy/dt + 2y = 0$, $y(0)=1$, $dy/dt=0$ ved $t=0$

```
>> dsolve('D2y-2*Dy+2*y=0','y(0)=1','Dy(0)=0')
ans =
-exp(t)*sin(t)+exp(t)*cos(t)
>>
```

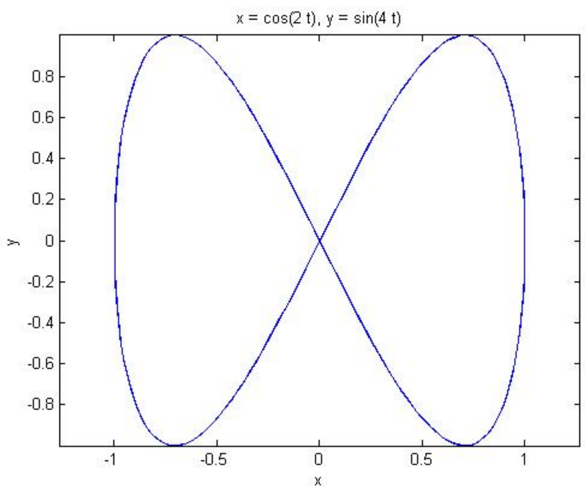
Som betyr:
$y = -e^t(\sin(t) - \cos(t))$

Plotting av symbolfunksjoner

Plotting av symboler med kommandoen **ezplot(S)** med
ezplot(S,[min,max]) for uavhengig variabel
ezplot(S,[xmin,xmax,ymin,ymax]
ezplot(S1,S2)
Straks plottet er laget kan det reformateres med plot og fplot.




```
>> A=(4*x+3)/(2*x-2)
A =
(4*x+3)/(2*x-2)
>> ezplot(A)
>>
```




```
>> x=cos(2*t)
x =
cos(2*t)
>> y=sin(4*t)
y =
sin(4*t)
>> ezplot(x,y)
<
```

Numeriske beregninger med symboluttrykk

Når man har laget et symbolsk uttrykk kan det være behov for å erstatte symbolvariable med tall og dette gjøres med kommandoen **subs**. Hvis man skal erstatte en symbolvariabel (var)i et symboluttrykk (S) med en numerisk verdi (tall)

R=subs(S,var,tall)

Tallet kan være ett tall (skalar), en vektor eller en matrise.

Eksempel:

La $S=0.5x^3+2e^{(0.6x)}$

S=0.5*x^3+2*exp(0.6*x)

```
>> syms x
>> S=0.5*x^3+2*exp(0.6*x)
S =
1/2*x^3+2*exp(3/5*x)
>> SD=diff(S)
SD =
3/2*x^2+6/5*exp(3/5*x)
>> %Tar den deriverte av S
>> subs(SD,x,2)
ans =
    9.9841
>> %Erstatter x=2 i SD
```

```
>> SDU=subs(SD,x,[1:0.5:8])
SDU =
  Columns 1 through 5
    3.6865    6.3265    9.9841   14.7530   20.7596
  Columns 6 through 10
   28.1744   37.2278   48.2307   61.6026   77.9102
  Columns 11 through 15
   97.9179  122.6579  153.5236  192.3956  241.8125
>>
```

Man kan også erstatte to eller flere symbolvariable med tall

R=subs(S,{var1,var2},{tall1,tall2})

```
>> syms a b c e x
>> S=a*x^e+b*x+c
S =
a*x^e+b*x+c
```

%Lager ut symboluttrykk ax^e+bx+c

Deretter alle symbolvariable med skalarer (tall)

```
>> subs(S,{a,b,c,e,x},{5,4,-20,2,3})
ans =
    37
>> T=subs(S,{a,b,c},{6,5,7})
T =
6*x^e+5*x+7
>> R=subs(S,{b,c,e},{[2 4 6],9,[1 3 5]})
R =
[   a*x+2*x+9, a*x^3+4*x+9, a*x^5+6*x+9]
>> %Resultatet er en vektor som symboluttrykk
>> W=subs(S,{a,b,c,e,x},{[4 2 0],[2 4 6],[2 2 2],[1 3 5],[3 2 1]})
W =
    20    26     8
>> %som er en vektor med numeriske verdier
```

Eksempel

Antall mengden legemiddel M i kroppen avhenger av hvor mye som tas opp i kroppen og hvor mye som tilføres

dM/dt=-kM+p

Hvor k er en proporsjonalitetsfaktor og p er hastigheten som medikamentet tilføres kroppen.

1.Bestem k hvis halveringstiden for medisinen er 3 timer ?

2. Det tilføres 50 mg/time, hvor det ikke var noe medisin fra t=0Lag et uttrykk for M som funksjon av tid og plott dette

Løsning:

Poroporsjonalitetskonstaten kan bestemmes ut fra hvor medisinen tas opp og det ikke tilføres noe mer

dM/dt=-kM som kan løses ved initialbetingelsene M=M0 ved t=0

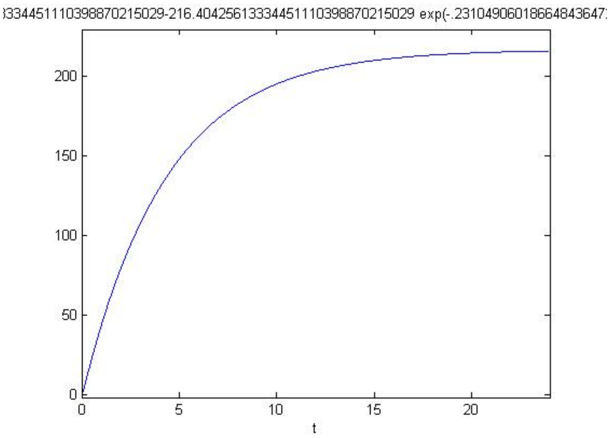
```
>> syms M M0 k t
>> Mt=dsolve('DM=-k*M','M(0)=M0')
Mt =
M0*exp(-k*t)
```

>>%Som betyr at $M(t)=M0e^{-kt}$. Halveringstid 3 timer betyr at ved t=3 er M(t)=1/2M0
>>%Løser: $0.5=e^{-3k}$.

```
>> ks=solve('0.5=exp(-k*3)')
ks =
.23104906018664843647241070715272
```

I det videre er differensialligningen dM/dt=-kM+p , k har vi bestemt og p=50mg/time.
Løser først differensialligningen:
>>%Løser dM/dt=-kM+p

```
>> syms p
>> Mtb=dsolve('DM=-k*M+p','M(0)=0')
Mtb =
p/k-p/k*exp(-k*t)
>> %Lager nå et plot hvor 0<t<24 timer
>> pgitt=50;
>> Mtt=subs(Mtb,{p,k},{pgitt,ks})
Mtt =
216.40425613334451110398870215029-
216.40425613334451110398870215029*exp(-
.23104906018664843647241070715272*t)
>> ezplot(Mtt,[0,24])
>>
```



Populasjonsdynamikk

Anta vi starter med en populasjon med størrelse P0. Populasjonen etter n tidsenheter er Pn. Anta at populasjonen øker eller minsker med en fast andel

$P_{n+1}=P_n + rP_n$

Hvor r er forskjellen mellom fødselsrate og dødsrate.

$(P_{n+1}-P_n)/P_n = r$

La oss ta med et ledd som senker veksten

$(P_{n+1}-P_n)/P_n = r - uP_n$

Vi lar for enkelhet antal u=1+r
Og vi får:
$P_{n+1}=uP_n(1-P_n)$
Hvor u er en positiv konstant og hvor populasjonen går mellom 0-1 som tolkes som % maksimum:
```
>> f=@(x,u)u*x*(1-x);
>> Xinit=0.5,X=itseq(f,Xinit,20,0.5),plot(X)
```

```
function x=popul(f,xinit,n,r)
%Beregner en iterativ sekvens av verdier
xinit=100
r=0.1
n=10
f=@(x,r) x*(1+r);
x=zeros(n+1,1);
x(1)=xinit;
for k=1:n
    x(k+1)=f(x(k),r);
end
```

Populasjonsdynamikk

Veksten til en populasjon kan modelleres som en differensialligning. Differensialligning for logistisk vekst av populasjonen x som funksjon av t:

(1) $dx/dt=x(1-x)=x-x^2$

hvor x er en fraksjon av maksimal populasjon. Denne kan løses med dsolve:
```
>> dsolve('Dx=x-x^2')
ans =
1/(1+exp(-t)*C1)
>> syms x0,sol=dsolve('Dx=x-x^2','x(0)=x0')
sol =
1/(1-exp(-t)*(-1+x0)/x0)
```

Denne inneholder 0 løsning så en bedre løsning blir:
```
>> syms x0,sol=dsolve('Dx=x-x^2','x(0)=x0')
sol =
1/(1-exp(-t)*(-1+x0)/x0)
>> bettersol=simplify(sol)
bettersol =
-x0/(-x0-exp(-t)+exp(-t)*x0)
>> subs(bettersol,x0,0)
ans =
0
```

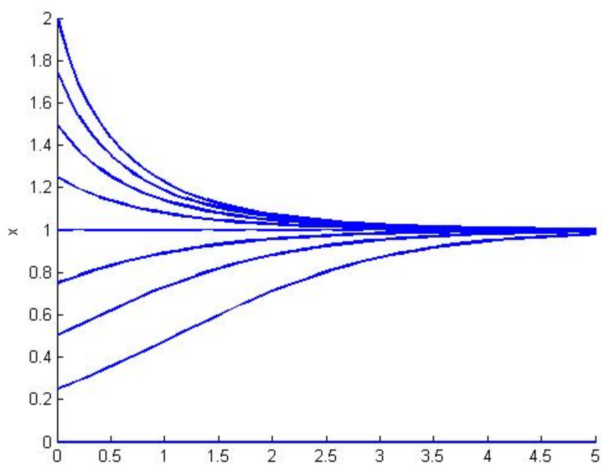
Anta at den intitelle verdien av x_0=x(0)og bruk verdiene x_0=0, 0.25,….2 og lag grafisk fremstillingen og uansett når x0>0 så vil x(t) bli lik 1 i det lange løp:

Den logistiske modellen har følgende underliggende prinsipper
Ideelt vil populasjonen øke proporsjonalt med den nåværende total dvs eksponensiell vekst, dette tilsvarer x leddet i (1)
Og fordi det er interaksjon mellom flere arter som naturlig begrenser veksten så vil ubegrenset eksponensiell vekst holdes i sjakk av leddet $-x^2$ i (1).
Anta at vi har to arter x(t) og y(t) som konkurrerer om samme ressurs for å overleve. Derved vil det bli ytterligere ett negativt ledd i differensialligningen som reflekterer interaksjon mellom artene. Den vanlige modellen antar at dette er proporsjonalt med produktet av de to populasjonene og desto større proporsjonalitetskonstant, jo mer alvorlig interaksjon

(2) $dx/dt=x-x^2 -0.5xy$
$dy/dt=x-x^2-0.5xy$

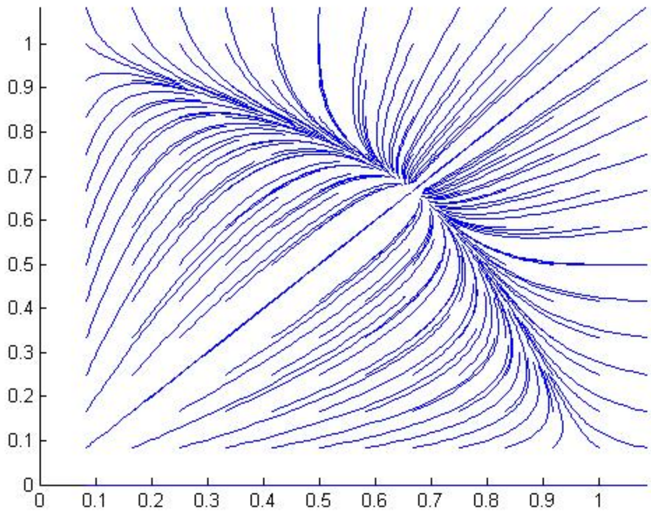
dsolve kan løse mange ordinære differensialligninger, spesielt lineære, men en blanding av kvadratledd så lar den seg ikke løse symbolsk, men det kan gjøres numerisk med ode-solver hvor initielle verdier er:
x(0)=0:1/12:13/12
y(0)=0:1/12:13/12

```
>> t=0:0.1:5;
>> cla reset;hold on
>> solcurves=@(t,x0)eval(vectorize(bettersol));
>> for interval=0:0.25:2
 plot(t,solcurves(t,interval),'LineWidth',1.5)
 end
>> axis tight
>> ylabel('x')
>>hold off
```




```
>> cla reset,hold on
```

```
>> f=@(t,x)[x(1)-x(1)^2-0.5*x(1)*x(2);...
x(2)-x(2)^2-0.5*x(1)*x(2)];
>> for a=0:1/12:13/12
for b=0:1/12:13/12
[t,xa]=ode45(f,[0 3],[a,b]);
plot(xa(:,1),xa(:,2))
end
end
>> axis([0 13/12 0 13/12]);hold off
```



Denne figuren er et faseportrett (phase portrait) av systemet. Ved bruk av ode45 skrives differensialligningen som en enkelt ligning med variabel x. Dens to komponenter representerer populasjonene x og y,. Endepunktene på kurvene er startpunkter, slik at enhver kurve som starter i første kvadrat, dette er det som tilsvarer situasjonen hvor begge populasjonene er tilstede ved starten, tenderer mot punktet (2/3,2/3). Ved x=y=2/3 forsvinner høyre ledd i ligning (2) og den deriverte er 0 og verdiene x(t) og y(t) blir konstante, dvs. de avhenger ikke av t. Hvis bare en art er tilstede fra begynnelsen, dvs. du starter ved en av eksene, så vil løsningen tendere mot enten 81,0) eller (0,1) avhengig om enten det er arten x eller y som er tilstede. Det var det samme som man så i forrige figur. Denne funksjonen tilsvarer fredelig sameksistens mellom de to artene, mens den nedenfor tilsvarer dommedag, hvor 2 indikerer sterk interaksjon og konkurranse mellom de to artene.

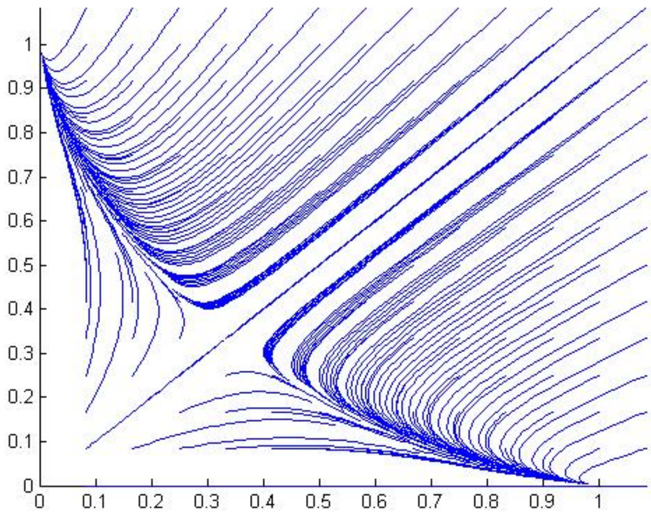
Hvis vi nå erstatter 0.5 med 2

(3) $dx/dt=x-x^2 -2xy$

$dy/dt=x-x^2-2xy$

```
>> cla reset,hold on
>> f=@(t,x)[x(1)-x(1)^2-2*x(1)*x(2);...
x(2)-x(2)^2-2*x(1)*x(2)];
```

```
>> for a=0:1/12:13/12
for b=0:1/12:13/12
[t,xa]=ode45(f,[0 3],[a,b]);
plot(xa(:,1),xa(:,2))
end
end
>> axis([0 13/12 0 13/12]);hold off
```



Hvor begge kurvene tenderer mot et av punktene 1,0 eller 0,1, spesielt or en løsningskurve som starter på en av aksene, som sier at det ikke er noen initiell populasjon for den andre arten. Uansett hvilken populasjon som starter med den største populasjonen som vil den ta over populasjonen og den andre vil dø ut.

Fraktaler

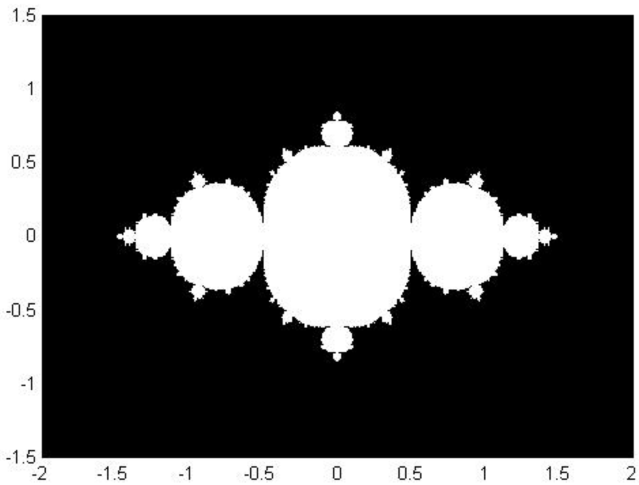
Anta at du starter med en rekke med komplekse tall fra z_0 slik at $z_{n+1}=z_n^2-0.75$
For noen verdier av z_0 vil sekvensen av tall divergere mot uendelig når n øker, men for andre verdier vil sekvensen holde seg innenfor grenser. Grensen av verdier for z0 kalles Julia-mengden $f(z)=z^2-0.75$
Bruk image og imagesc for å tegne Julia-mengden. Reele verdier for z0 ligger mellom -2 og 2, og imaginære ligger mellom -1.5 og 1.5. Lag en grid med z-verdier i et rektangel for eksempel 300x400 og beregn hver z1, z2, zn for n

```
xvals=linspace(-2,2,400);
yvals=linspace(-1.5,1.5,300);
[x,y]=meshgrid(xvals,yvals);
z=x+i*y;
for k=1:100
z=z.^2-0.75;
end
```

```
clf reset
set(gcf,'Color','Red')
imagesc(xvals, yvals,isfinite(z))
colormap(yellow)
set(gca,'YDir','normal')
axis equal tight
```

Hvis denne istedet:

```
>> clf reset
>> set(gcf,'Color','White')
>> >> set(gcf,'Color','White')
>> colormap(gray)
>> set(gca,'YDir','normal')
>> exis equal tight
```



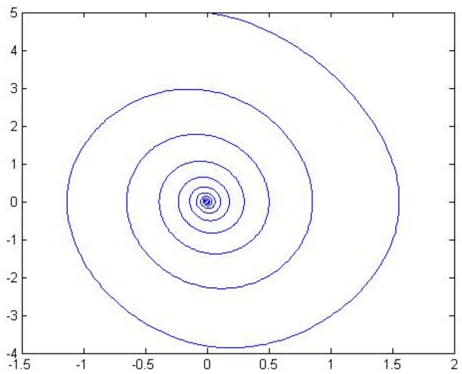
Kommandoen isfinite brukes til å sette verdien 1 for verdier i arrayed som er endelige flyttall og 0 for de som har divergrt. Bruker set for å versere vertikalakse.

```
xvals=linspace(-2,2,400);
yvals=linspace(-1.5,1.5,300);
[x,y]=meshgrid(xvals,yvals);
z=x+i*y;
for k=1:100
    z=z.^2-0.75;
end
clf reset
set(gcf,'Color','Yellow')
imagesc(xvals,yvals,isfinite(z))
colormap(cyan)
set(gca,'YDir','normal')
axis equal tight
```

360^{o}pendel

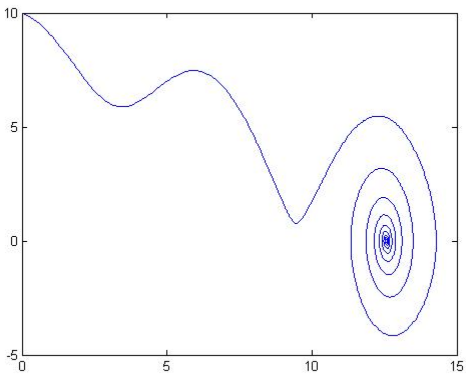
En pendel med lengde 1 m og massen av pendelen er 1 kg. T er tiden i sekunder og x er vinkelen pendelen har fra vertikal posisjon i radianer, og y er vinkelhastigheten for pendelen i radianer per sekund, og friksjonskoeffisienten er 0.5
Formelen for pendelbevegelsen
dx/dt= y(t)
dy/dt=-0.5y(t) - 9.81sin(x(t))
Bruker ode45 til numerisk løsning hvor x og y kombineres i en vektor x :

```
>> g=@(t,x)[x(2);-0.5*x(2)-9.81*sin(x(1))];
>> [t xa]=ode45(g,[0:0.01:20],[0 5]);
>> plot(xa(:,1),xa(:,2))
```



Starter ved (0,5) og når t øker følger vi spiral mot (0,0). Vinkelen oscillerer fram og tilbake, med stadig mindre pendelbevegelse og står stille ved t=20
Hvis inititalhastigheten øker til 10:

```
>> [t xa]=ode45(g,[0:0.01:20],[0 10]);
>> plot(xa(:,1),xa(:,2))
```



Tidsvinkel øker til over 14 radianer før kurven danner en spiral ved (12.5 0), dvs. mot (4π,0) hvor 4πradianer er den samme posisjon som 0 radianer

Numerisk løsning av varmeligningen

Varmeligningen og diffusjonsligningen kan løses numerisk. Dette er en partiell differensiallikning for varmekonduksjon eller diffusjon. I 3D er varmeligningen:

$$\frac{\partial u}{\partial t} = k\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

Hvor u er funksjon av t, x, y og z og representerer temperatur eller konsentrasjon ved diffusjon tiden t og posisjon (x,y,z), og k er en konstant avhengig av materialet som brukes og kalles termisk konduktivitet hvis varmestrøm og diffusjonskoeffisient hvis diffusjon. For enkelhets skyld anta at mediet er endimensjonalt, dvs. diffusjon i et vannfylt rør eller varmeflyt i en tynn isolert tråd. I dette tilfellet blir den partielle differensialligningen:

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 t}{\partial x^2}$$

Hvor u(x,t) er temperatur eller konsentrasjon ved tid t og avstand x

```
function u=heat(k,t,x,init,bdry)
J=length(x);
N=length(t);
dx=mean(diff(x));
dt=mean(diff(t));
s=k*dt/dx^2;
u=zeros(N,J);
u(1,:) = init;
for n=1:N-1
    u(n+1,2:J-1)=s*(u(n,3:J)+u(n,1:J-2))+...
        (1-2*s)*u(n,2:J-1);
    u(n+1,1)=bdry(1);
    u(n+1,J)=bdry(2);
end
tvals=linspace(0,4,41);
xvals=linspace(-5,5,21);
init=20 + 5*sign(xvals);
uvals=heat(2,tvals,xvals,init,[15 25]);
surf(xvals, tvals, uvals)
xlabel x; ylabel t; zlabel u
```

Litteratur:

Gilat, Amos: MATLAB. An introduction with applications. 2E. John Wiley & Sons,Inc.2005

Hunt, B.R.; Lipsman, R.L. & Rosenberg, J.M.: A guide to Matlab for beginners and experienced users. 2E. Cambridge University Press 2006