Kapitel 0

Datorövning i MATLAB

Här skall vi bekanta oss med beräkningsprogrammet MATLAB, som kommer att vara vårt verktyg för datorarbete.

Datorövningen är uppdelad i ett antal sektioner där olika aspekter av MATLAB behandlas. Varje sektion inleds med en kort beskrivning och därefter ges exempel på hur programmet kan användas. Referenser till "Pärt, E. Användarhandledning i MATLAB" ges enligt –se Pärt, 2.5.

Exemplen innehåller dialoger med datorn. Det som skrivs med `denna stil` är precis som det Du ser på Din datorskärm när Du genomför exemplen. Insprängt i exemplen finns kommentarer och förtydliganden skrivna med vanlig stil.

Tecknet » är MATLABS prompter. Den text som följer efter promptern är min inmatning. Systemets svar följer på raden (eller raderna) efter. Om Du gör ett fel när du försöker genomföra exemplen, så talar datorn om att något är fel. Då är det bara att försöka igen!

0.1 MATLAB som kalkylator

MATLAB kan användas som en avancerad kalkylator. Man knappar in matematiska uttryck som man skriver dem på papper och får svaret direkt. Nästan alla matematiska funktioner finns tillgängliga –se Pärt, 2.5. Man kan använda parenteser till godtyckligt djup.

Inmatning på en rad avslutas alltid med $\boxed{\text{Enter}}$-tangenten.

Gör några enkla beräkningar. Multiplikation och division görs med `*` respektive `/`.

De trigonometriska funktionerna arbetar med radianer.

Kvadratroten ur `tal` skrivs `sqrt(tal)`.

```
>> 3+2-1

ans =

     4
```

Gör ytterligare några enkla beräkningar.

```
>> 2^6-1

ans =
    63

>> cos(pi/4)-sqrt(2)/2

ans =
     0
```

- Storheter (variabler) kan tilldelas godtyckliga namn. Dessa namn finns sedan kvar i systemet ända tills man går ur MATLAB eller suddar ut dem.
- Om man skriver en variabels namn, så skrivs dess värde ut.
- Om en rad avslutas med semikolon (;) undertrycks utskriften.
- Om man gör en beräkning utan att tilldela resultatet ett namn (d.v.s. utan att ha ett vänsterled), så hamnar resultatet i variabeln `ans`.

```
>> x=2

x =

     2

>> x=5

x =

     5

>> y=x/25

y =

    0.2000

>> y=x/25;
```

```
>> y

y =

    0.2000

>> mitt_resultat=sin(y*pi);
>> mitt_resultat

mitt_resultat =

    0.5878

>> exp(mitt_resultat^5)

ans =

    1.0727
```

Vi har redan använt några av funktionerna i MATLAB, t.ex. `sqrt, sin, cos` och `exp`. Om Du skriver `help` följt av namnet på en funktion, så får Du information om funktionen.

Skriv ut hjälptexten för funktionen `sqrt` .

```
>> help sqrt

 SQRT Square root.
  SQRT(X) is the square root of X.
        Complex results are produced if
        X is not positive.

  See also SQRTM.
```

Alla variabler Du skapar läggs på den arbetsyta (*workspace*), som Du arbetar på.

- Arbetsytor kan sparas (`save`) för att senare återladdas (`load`). Då en arbetsyta återladdas befinner man sig i samma läge som då den sparades, d.v.s. alla variablerna har samma värden som innan man avslutade föregående arbetspass vid datorn.
- Med `who` får man reda på vilka variabler man har för tillfället.
- Med `clear` suddas hela arbetsytan. Se –Pärt, 2.4 sid 29.
- Piltangenterna kan användas till att bläddra i gamla inmatade rader, ändra i dem och låta datorn utföra det som nu står på raden.

```
>> who

Your variables are:

ans                 x
mitt_resultat       y

>> save temp
>> clear
```

```
>> who

Your variables are:


>> x
??? Undefined function or variable x.
```

Arbetsytan är tom. Inga variabler finns definierade. Vi kan nu lämna MATLAB och logga ut från maskinen. Då vi loggar in igen kan vi fortsätta enligt nedan. Vår arbetsyta är permanent lagrad i datorn och kan återlagras när helst vi vill. Vi fortsätter exemplet (utan att lämna MATLAB emellan) och återlagrar arbetsytan temp.

```
>> load temp
>> x

x =
```

```
     5

>> who

Your variables are:
```

```
ans              x
mitt_resultat    y
```

De fyra variablerna är återigen tillgängliga, med sina gamla värden.

0.2 Vektorer

Kolumnvektorn

$$\mathbf{x} = \begin{pmatrix} 5 \\ 4 \\ 3 \end{pmatrix}$$

kan matas in till datorn på något av följande sätt

```
>> x=[5
      4
      3]
```

```
>> x=[5;4;3]
```

```
>> x=[5 4 3]'
```

I samtliga fall svarar datorn med

```
x =

     5
     4
     3
```

I det första inmatningsexemplet stegar man fram 5 steg med mellanslagstangenten (den långa) på raderna två och tre för att få lättläst inmatning. Inmatningsfas där vänster hakparantes (`[`) matats in kan avslutas först då en höger hakparantes (`]`) matats in.

Semikolon (`;`) används i vektorer till att markera ny rad. I det tredje inmatningsexemplet skapas en radvektor $(5 \quad 4 \quad 3)$ som därefter transponeras. Transponeringsoperatorn ^{T} skrivs i MATLAB som '.

Exempel 1 Mata in vektorerna **u** och **v** där

$$\mathbf{u} = \begin{pmatrix} 2 \\ -5 \\ 4 \\ 2 \end{pmatrix} \qquad \mathbf{v} = \begin{pmatrix} 1 \\ 1 \\ -2 \\ 4 \end{pmatrix}$$

och beräkna

a. $\mathbf{u} + \mathbf{v}$

b. $\mathbf{u} - 3\mathbf{v}$

c. $\mathbf{u}^T$

d. $\mathbf{u}^T\mathbf{v}$

e. $(\mathbf{u} - 3\mathbf{v})^T(\mathbf{u} + 3\mathbf{v})$

Gör de ovan angivna beräkningarna med dator och för hand.

Lösning Dialogen blir

```
>> u=[2;-5;4;2]

u =

     2
    -5
     4
     2

>> v=[1;1;-2;4]

v =
```

```
      1
      1
     -2
      4

>> u+v           %a.

ans =

      3
     -4
      2
      6

>> z=u-3*v       %b.

z =

     -1
     -8
     10
    -10

>> u'            %c.

ans =

     2    -5     4     2

>> u'*v          %d.

ans =

    -3

>> z'*(u+3*v)    %e.

ans =

  -149
```

Observera att resultatet i b. lagras i en kolumnvektor `z`. Den vektorn har därefter använts i beräkningarna i e. För de övriga delproblemen beräknas resultaten och lagras i den tillfälliga variabeln `ans`. Notera t.ex. att `ans` är en kolumnvektor i a, en radvektor i c. och en skalär i d.

Addition och subtraktion av vektorer kan endast utföras på vektorer med samma antal komponenter (samma dimension). Det går inte heller att addera en radvektor och en kolumnvektor.

```
>> x=[1 2 4 6]'; z=[2 1]';
>> x+z
??? Error using ==> +
Matrix dimensions must
agree.

>> y=[1 2 4 6]
y =
     1     2     4     6
```

```
>> x+y
??? Error using ==> +
Matrix dimensions must agree.
```

Skriv `x` resp. `z` så Du kan inspektera dessa vektorer.

0.3 Funktioner och vektorer

En typ av vektorer skapar man genom att tabellera givna funktioner $f(x)$. Definiera först för vilka x-värden funktionen skall tabelleras. För att ange en följd av värden används kolon (:). Se exemplen nedan för detaljer. Då erhålls en vektor **x**. Hela denna vektor kan sedan vara argument till alla matematiska funktioner i MATLAB. För att göra multiplikation, division, exponentiering mm. komponent för komponent sätter man en punkt (.) före operationen. Addition och subtraktion utförs naturligtvis alltid komponentvis.

Vi arbetar nedan med radvektorer för att få rum med många exempel på en sida.

```
>> t=1.5:0.5:2.5

t =
    1.5000    2.0000    2.5000


>> x=0:2:6

x =
     0     2     4     6

  >> x.^2

  ans =
       0     4    16    36
```

```
>> x+x.^2

ans =
     0     6    20    42

>> z=x+x.^2

z =
     0     6    20    42

>> f=2-2*x+3*x.^2+x.^3

f =
     2    18   106   314
```

Den sista satsen beräknar polynomet $f(x) = 2 - 2x + 3x^2 + x^3$. Vi får direkt en vektor av f-värden svarande mot de x-värden som finns i vektorn **x**. Observera att tvåan direkt efter likhetstecknet inte är en vektor, men då en skalär storhet adderas till/subtraheras från en vektor så omvandlas skalären till en vektor av samma dimension som vektorn.

Man kan inte dividera med 0. I exemplet nedan försöker datorn räkna ut 0/0 och redovisar resultatet som NaN (Not a Number). Första komponenten av resultatet är odefinierad, men de övriga komponeterna har beräknats korrekt. I det andra exemplet har vi använt konstanten `pi`$=\pi$.

```
>> z./x
Warning: Divide by zero

ans =

   NaN     3     5     7


  >> g=sin(pi*x/12)

  g =

         0    0.5000    0.8660    1.0000
```

Då en vektor är argument till en funktion, som t.ex. `sin` , så blir resultatet en vektor med samma dimension som argumentet. Motsvarande gäller då argumentet är en matris.

0.4 Matriser

Matrisen

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 5 & 4 & 9 \\ 8 & 7 & 6 \end{pmatrix}$$

kan matas in på något av följande sätt

```
>> A=[1 2 3
      5 4 9
      8 7 6]

>> A=[1 2 3; 5 4 9; 8 7 6]
```

I båda fallen erhålls på skärmen utskriften

```
A =

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

Semikolon (;) används i matriser till att markera slut på raden.

En matris kan också bildas genom att vi ställer samman vektorer.

Exempel Definiera tre kolumnvektorer **u**, **v**, **z** med fyra komponenter och ställ sedan samman dem till en matris B, med första kolumnen=**u**, andra kolumnen=**v** och tredje kolumnen=**z**. B får storleken (eller är av typ) 4×3.

```
>> u=[1:1:4]'

u =

     1
     2
     3
     4

>> v=[2 2 1 1]'
```

```
v =

     2
     2
     1
     1

>> z=v.^2
z =

     4
     4
     1
     1

>> B=[u v z]

B =

     1     2     4
     2     2     4
     3     1     1
     4     1     1
```

Matriser kan analogt byggas upp av radvektorer. Med satsen
`»C=[u'; v'; z']`
skapas en matris. Vilken?

Vi kan referera till ett element av matrisen genom att ange radindex och kolumnindex.

```
>>B(3,1)
ans =
     3
```

Vi använder paranteserna `( )` då vi *refererar* till ett element, medan `[ ]` används då en matris skapas! Ett elements värde kan ändras med en tilldelning. Hela matrisen skrivs ut efter ändringen.

```
>> B(2,2)=12
B =

     1     2     4
     2    12     4
     3     1     1
     4     1     1
```

Det finns ytterligare ett stort antal sätt att skapa matriser, som vi skall återkomma till allt eftersom vi behöver dem –se Pärt 4.1-4.4.

Observera att vektorer kan uppfattas som specialfall av matriser.

- en kolumnvektor med dimensionen n är en matris av typ $n \times 1$,
- en radvektor av dimensionen m är en matris av typ $1 \times m$,

Vi skall nu öva på några av räknereglerna för matriser och vektorer. Beräkningarna nedan bör göras både för hand och med dator för att nöta in räknereglerna. Under den lärarledda övningen kanske Du inte gör alla uppgifterna, och kanske Du spar handräknandet till något annat tillfälle. Gör så många uppgifter att Du känner dig säker på att klara av att självständigt göra de övriga.

I exemplen arbetar vi med små matriser med heltaliga element för att få rum med dem på papper och lätt kunna kontrollräkna för hand eller med kalkylator. Allt det som görs gäller naturligtvis även för stora matriser med element som inte är heltal. Elementen kan även vara komplexa tal.

Vi definierar två st 2×2 matriser A och B, och två kolumnvektor $\mathbf{x}$ och $\mathbf{y}$ med 2 komponenter.

```
>> A=[2 1
      4 5]

A =

     2     1
     4     5

>> B=[1 1
      2 3]

B =

     1     1
     2     3

>> x=[7; 9]

x =

     7
     9

>> y=[-6; 1];
```

Addera och subtrahera, multiplicera med skalär, och multiplicera matris och vektor

```
>> A+B

ans =

     3     2
     6     8

>>2*B

ans =

     2     2
     4     6

>> A-2*B

ans =

     0    -1
     0    -1

>> A*x

ans =

    23
    73
```

Formulera och lös det linjära ekvationssystemet

$$\begin{pmatrix} 2 & 1 \\ 4 & 5 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 23 \\ 73 \end{pmatrix}$$

```
>> A=[2 1; 4 5]

A =

     2     1
     4     5

>> b=[23; 73]

b =

    23
    73

>> x=A\b

x =

     7
     9


>> %Kontrollräkning
>> A*x

ans =

    23
    73
```

Gör några beräkningar för att verifiera att $A(5\mathbf{x}) = 5A\mathbf{x}$ och $A(\mathbf{x}+\mathbf{y}) = A\mathbf{x} + A\mathbf{y}$

```
>> z=5*x; w=A*x;

>> A*z-5*w

ans =

     0
     0

>> z=x+y

z =

     1
    10

>> A*z

ans =

    12
    54

>> A*x+A*y

ans =

    12
    54
```

0.5 Grafik

Två viktiga fönster i Matlab är:

- kommandofönstret
- grafikfönstret

De exempel vi sett hittills har endast använt kommandofönstret. Kurvor, grafer och andra figurer hamnar i grafikfönstret. Vi kan rita xy-grafer, 3-dimensionella grafer, mm. Kommandot

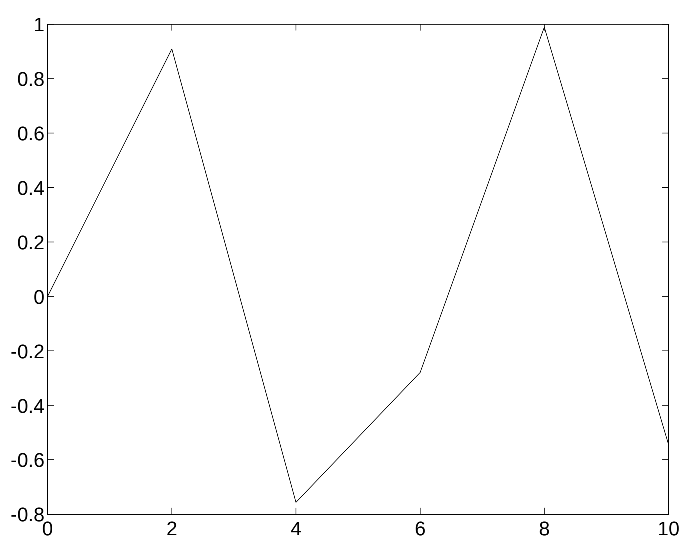
`plot`

används för 2-dimensionell grafik. Se kommandot nedan för ett exempel. Första argumentet representerar x-värden (horisontella axeln) och det andra y-värden (vertikala axeln). Man återgår till kommandofönstret genom att klicka på det eller genom att trycka på en godtycklig tangent. Man går till grafikfönstret och senast ritade bild genom kommandot `shg` (show graphics). Grafikfönstret suddas med `clf`.

Skapa en radvektor **t** = (0 2 4 6 8 10), d.v.s. komponenterna skall vara från 0 till 10 med steget 2. Rita sinus av vektorn **t** (y-led) mot vektorn **t** (x-led).

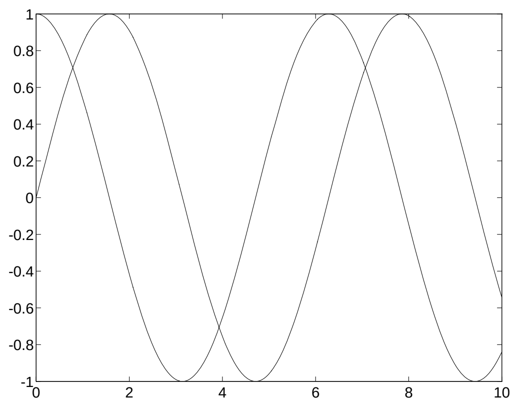
```
>> t=0:2:10;
>> plot(t,sin(t))
```

Grafen ser inte ut som en sinuskurva. Vi har så få tabellvärden att figuren inte blir rättvisande, ty mellan tabellvärden dras räta linjer. Vi behöver många fler värden för att få en snygg sinuskurva.



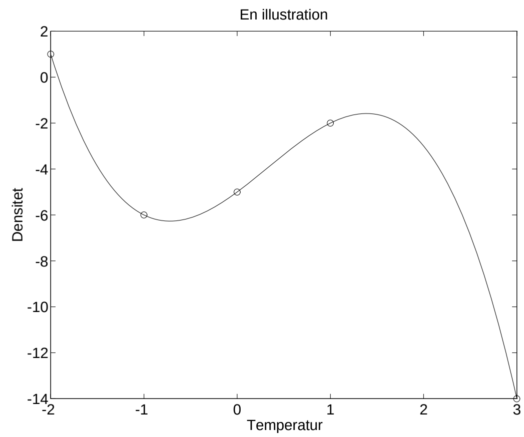
Skapa vektorn **t** med komponenter från 0 till 10 med steget 0.01 och rita sinus av t och cosinus av t. Observera att vektorerna nu har ca 1000 komponenter. En vektors *dimension* (=antal komponenter) kan vi bestämma med `length`.

```
>> t=0:0.01:10;
>> length(t)

ans =

        1001

>> plot(t,sin(t),t,cos(t))
```



Man kan sätta beteckningar på axlarna och rubrik över plot-bilden, se –Pärt, 13.4, sid 279. Om så önskas kan man välja att markera punkter och olika linjetyper, och olika färger på linjerna (förutsätter färggrafik), se –Pärt, 13.2, sid 260.

```
>> x=[-2 -1 0 1 3];
>> y=[1 -6 -5 -2 -14];
>> [x' y']

ans =

    -2     1
    -1    -6
     0    -5
     1    -2
     3   -14

>> t=-2:0.1:3;
>> f=-5+3*t+t.^2-t.^3;
>> plot(t,f,x,y,'o')
>> title('En illustration')
>> xlabel('Temperatur')
>> ylabel('Densitet')
```



I föregående exempel arbetade vi med radvektorer. För att klart se vilka x- och y-värden som hör ihop (komponent för komponent) använde vi `[x' y']`, d.v.s vektorena transponeras och ställs samman i en matris. I plotkommandot utgör de två första argumenten `t,f` en enhet, och de återstående `x,y,'o'` en andra enhet. Det sista argumentet här anger att för denna kurva skall endast punkter markerade med `o` ritas ut. Man kan i samma plotkommando rita många kurvor.

0.6 Dagbok

När man arbetar med omfattande beräkningar kan det vara bra att kunna "komma ihåg" vilka in- och utmatningar som gjorts. För detta ändamål finns kommandot `diary` med vilket man kan föra protokoll (upprätta en dagbok) över allt som skrivs i kommandofönstret – se Pärt 2.7, sid 40. Dagboken lagras på en fil som man kan editera, inspektera, eller skriva ut på papper. Senare skall vi se att ett bra sätt att skapa ett MATLAB-program (en m-fil) är att skapa en dagboksfil och sedan redigera den.

Starta registrering på filen uppgift21.d. Gör några beräkningar och avsluta därefter registreringen.

```
>> diary uppgift21.d
>> x=[-2:0.02:1]';
>> x'*x

ans =

  152.5100

>> length(x)

ans =

   151

>> x(151)

ans =

     1

>> diary off
```

Dagboksfilen kan listas genom att från MATLAB ge kommandot `type`. Alternativt kan operativsystemets listkommando användas. Systemkommandon måste inledas med utropstecken (`!`) om de görs inuti MATLAB.

Skriv dagboksfilen `uppgift21.d` på skärmen

- `type ...`

```
>> type uppgift21.d
```

Följande rader t.o.m. diary off är innehållet i dagboksfilen

```
>> x=[-2:0.02:1]';
>> x'*x

ans =

  152.5100

>> length(x)

ans =

   151

>> x(151)

ans =
     1
>>diary off
```

Genom att skriva `diary on` återupptas registreringen på samma fil som användes tidigare. Det kan vara användbart ibland, men tag för vana att starta en ny dagboksfil i stället. Man kan mycket lätt drunkna i allt det skräp man sparar annars. Det går alltid att redigera i dagboksfilerna i efterhand; sudda sådant man inte vill spara, slå ihop flera dagboksfiler till en, lägga till förklarande text. Sessionsexemplen i denna skrift har skapats på detta sätt.

Era redovisningar av många av inlämningsuppgifterna skall göras med hjälp av dagboksfiler.

För att lätt hitta bland mina filer brukar jag alltid ge dagboksfiler efternamnet d (efternamn=*extension*), t.ex. lab2.d.

0.7 m-filer, kommandofiler och funktionsfiler

En följd av MATLAB-satser samlade i en fil med efternamnet m är en kommandofil. Denna fil kan man skapa med sin favoriteditor. Då förnamnet skrivs i kommandofönstret utförs de satser som finns i filen.

I filen `mfilex.m` finns följande MATLAB satser

```
x=0.24
tmp=(12-sqrt(x^2+(0.2/(2*x))));
z=[(2.3-exp(-x^2))/tmp
    0.2
    -4*x^(1/6)]
A=eye(3); B=ones(4);
```

Rensa arbetsytan och utför sedan satserna

```
>> clear

>> mfilex

x =

    0.2400


z =

    0.1199
    0.2000
   -3.1533
```

Observera att nu har t.ex. matrisen A definierats och har ett värde.

```
>> who

Your variables are:

A          B          x          z


>> A

A =

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

Kommandofiler kan användas för att

- skriva in stora uttryck som kan vara svåra att få rätt på en gång.
- definiera matriser med många rader och kolumner.
- skriva hela program.

Filerna kan editeras om det blev fel första gången.

Kommandofilen nedan heter `fack.m` och innehåller koefficientmatrisen och högerledet till ett linjärt ekvationssystem med 17 obekanta och 17 ekvationer.

Som Du kan förstå är det mycket lätt att göra misstag då matrisen ställs upp, så det är väsentligt att man lätt kan korrigera misstagen. Procenttecknet (%) används i MATLAB som inledning till en kommentar. Allt som står efter % på en rad är kommentar, d.v.s. datorn utför inte detta, utan det står där enbart för att förklara för den som läser texten. Detta exempel skall Du bara läsa igenom. Filen `fack.m` finns på disketten Du får ut, så Du kan leka med exemplet hemma. För att skriva och ändra programtext i separata filer måste Du behärska någon editor.

```
a=sqrt(2)/2;
% Kolumnnummer
%   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
%                                                       rad nod led
A=[-a  0  0  1  a  0  0  0  0  0  0  0  0  0  0  0  0      % 1  2   x
   -a  0 -1  0 -a  0  0  0  0  0  0  0  0  0  0  0  0      % 2  2   y
    0 -1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0      % 3  3   x
    0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0      % 4  3   y
    0  0  0 -1  0  0  0  1  0  0  0  0  0  0  0  0  0      % 5  4   x
    0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0      % 6  4   y
    0  0  0  0 -a -1  0  0  a  1  0  0  0  0  0  0  0      % 7  5   x
    0  0  0  0  a  0  1  0  a  0  0  0  0  0  0  0  0      % 8  5   y
    0  0  0  0  0  0  0 -1 -a  0  0  1  a  0  0  0  0      % 9  6   x
    0  0  0  0  0  0  0  0 -a  0 -1  0 -a  0  0  0  0      %10  6   y
    0  0  0  0  0  0  0  0  0 -1  0  0  0  1  0  0  0      %11  7   x
    0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0      %12  7   y
    0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  a  0      %13  8   x
    0  0  0  0  0  0  0  0  0  0  0  0  0 -1  a  0         %14  8   y
    0  0  0  0  0  0  0  0  0  0  0  0 -a -1  0  0  1      %15  9   x
    0  0  0  0  0  0  0  0  0  0  0  0  a  0  1  0  0      %16  9   y
    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -a -1];    %17 10   x

%   1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17

b= [0  0  0 10  0  0  0 10  0  0  0 10  0  0  0 10  0]';
```

Matrisen har så många kolumner att det är omöjligt att skriva ut matrisens samtliga element på ett överskådligt sätt. Vi använder därför kolon-notation för att definiera delmatriser av A –se Pärt 4.3, sid 74. Nedan utför vi filen `fack.m` och tittar därefter på vilka värden matriselementen i nedre högra hörnet har. Vi tittar på kolumnerna 12 till 17, raderna 8 till 17.

```
>> fack
>> A(8:17,12:17)

ans =

         0         0         0         0         0         0
    1.0000    0.7071         0         0         0         0
         0   -0.7071         0         0         0         0
         0         0    1.0000         0         0         0
         0         0         0         0         0         0
   -1.0000         0         0         0    0.7071         0
         0         0         0   -1.0000    0.7071         0
         0   -0.7071   -1.0000         0         0    1.0000
         0    0.7071         0    1.0000         0         0
         0         0         0         0   -0.7071   -1.0000
```

Ett sätt att skapa en m-fil är att spara sin dialog från kommandofönstret och därefter editera den. Se exemplet på nästa sida.

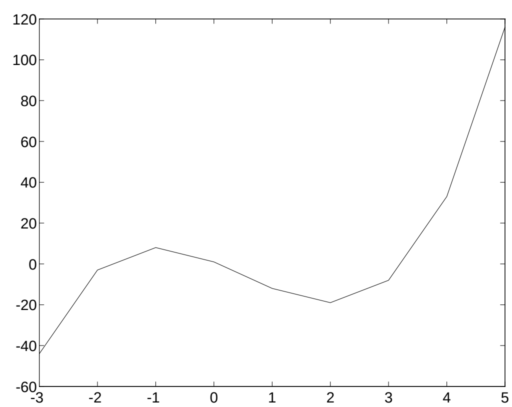
Funktioner returnerar ett eller flera värden, och kan ha en eller flera parametrar. Man kan skapa egna funktionsfiler, men vi kommer i denna kurs inte ha behov av egendefinierade funktioner, se Pärt 2.8 för mer information om behovet uppstår.

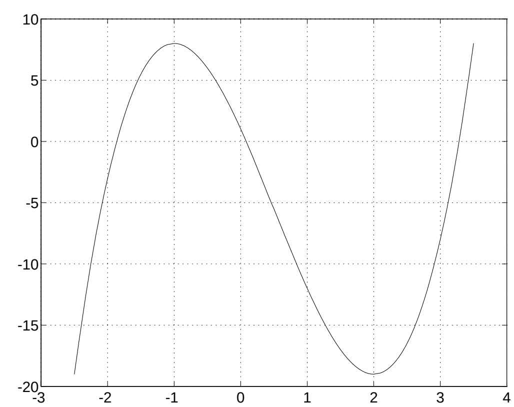
Vi skall rita kurvan $y = f(x)$ för $f(x) = 2x^3 - 3x^2 - 12x + 1$ och vill ha en snygg figur där minima och maxima syns klart.

```
>> diary ex46.m
>> x=-3:1:5;
>> f=2*x.^3-3*x.^2-12*x+1;
>> plot(x,f)
>> [x' f']

ans =

    -3   -44
    -2    -3
    -1     8
     0     1
     1   -12
     2   -19
     3    -8
     4    33
     5   116
```

```
>> x=-2.5:0.1:3.5;
>> f=2*x.^3-3*x.^2-12*x+1;
>> plot(x,f)
>> grid
```

När den första figuren ritats, ser man att den är för taggig, och att intervallet i x-led var onödigt stort. Vi valde därför ett mindre intervall, och tätare mellan x-värdena. Satsen där f beräknas för de nya x-värdena är identisk med den andra satsen i exemplet. Jag använde därför uppåt-pil tangenten till att återkalla den satsen (4 tryckningar på tangenten behövdes), och utförde den sedan genom att trycka på Enter-tangenten. Detsamma gjorde jag för plot-satsen.





I dagboksfilen `ex46.m` finns nu denna dialog. Där finns en blandning av satser och datorsvar, och promptern » finns också där.

Med min editor har jag sedan redigerat filen till:

```
x=-2.5:0.1:3.5;
f=2*x.^3-3*x.^2-12*x+1;
plot(x,f)
grid
disp('Klart!')
```

Observera särskilt att alla » är borttagna. Den sista raden har jag lagt till på skoj för att får ett meddelande om att filen är färdigbearbetad. I grafikfönstret finns nu den senare av figurerna på föregående sida.

Eftersom jag planerade att göra en m-fil av dagboken gav jag den efternamnet m i stället för d.

Du bör öva dig på att editera dina dagboksfiler på detta sätt. Under den lärarledda övningen hinner vi inte med det, och det är onödigt att Du lär dig den editor som vi använder på våra undervisningdatorer. När Du går igenom övningen på din egen dator, är det däremot viktigt att Du genomför även denna övning. Det har ingen betydelse vilken editor Du arbetar med, och Du behöver inte kunna editorn särskilt bra.

Övningsexempel:

1. Låt
$$A = \begin{pmatrix} 2 & 1 \\ 4 & 5 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} 7 \\ 9 \end{pmatrix}$$
och
   (a) Beräkna AB och BA.
   (b) Beräkna $\mathbf{z} = A\mathbf{x}$ och $\mathbf{z}^T\mathbf{z}$.
   (c) Beräkna $C = A^TA$ och $\mathbf{x}^TC\mathbf{x}$.

2. Definiera en matris E (enhetsmatrisen) med ettor på diagonalen och nollor för övrigt. Matrisen skall vara av typ 4×4.

3. Se Pärt 4.1 och definiera följande 5×5 matriser
   (a) E, enhetsmatris.
   (b) Ett, matris med alla element ettor.
   (c) Noll, matris med alla element nollor.

4. Lös det linjära ekvationssystemet
$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 4 & 16 & 64 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ -2 \\ 14 \end{pmatrix}$$

5. Rita grafen för funktionen $g(x) = x^3/20 - 2 - x^3e^{-x}$ för x på intervallet $(0, 5)$. Välj lämpligt tabellsteg så grafen blir snygg.

6. Rita i ett diagram vattenståndet som funktion av tiden t vid Lands End. Vattenståndet i meter ges av funktionen
$$h(t) = 2.1\sin(\frac{2\pi t}{30}) + 0.4\cos(\frac{2\pi t}{200})$$
där t är i dagar.
   (a) Rita h för t från 0 till 720.
   (b) Rita en ny figur med såväl h som den första termen och andra termen i uttrycket för h. Du skall alltså i samma figur rita tre kurvor. Sätt beteckningar på axlarna och rubrik.