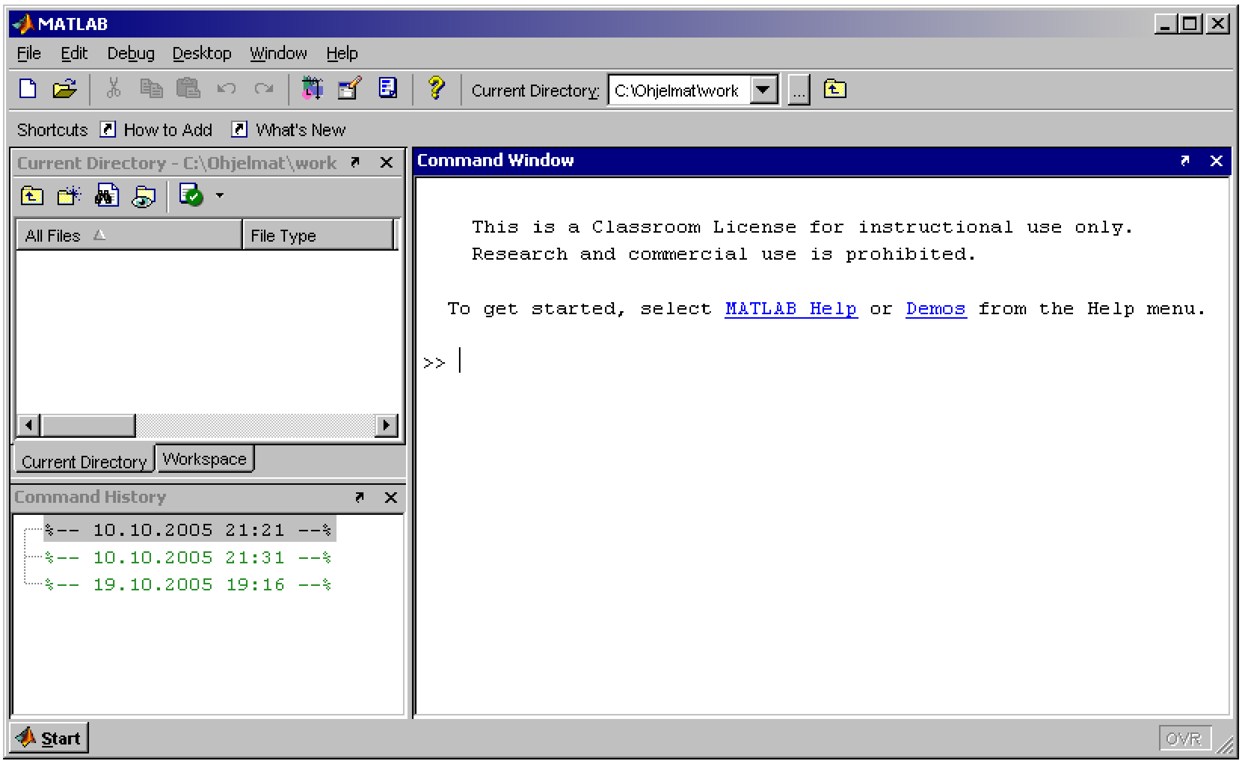
MATLAB 7.1
Komentoikkunaharjoitus

© Matti Lähteenmäki
2005
www.tamk.fi/~mlahteen/

SISÄLLYSLUETTELO

1 Aloitus ja ohjetoiminnot

Käynnistä MATLAB työpöydän kuvakeeesta tai valitsemalla Start-valikosta MATLAB. Esiin tulee MATLAB-työpöytä, jossa on telakoituna joukko ikkunoita ja mahdollisesti työpöydän ulkopuolella kelluvia ikkunoita. Avautuva ikkunavalikoima riippuu siitä, mitkä asetukset edellisellä käyttökerralla on jätetty voimaan. Siirry kuvan mukaisen standardityöpöydän käyttöön valitsemalla työpöydän valikosta Desktop > Desktop Layout > Default. Työpöydän oikeassa reunassa on komentoikkuna (Command Window), jossa näkyvän kehotteen >> perään käyttäjä kirjoittaa MATLABille antamansa komennot. Muut standardityöpöydän ikkunat ovat muuttuja-avaruus (Workspace), joka näyttää komentoikkunassa määritellyt vakiot ja muuttujat, komentohistoria (Command History) on lista aikaisemmin annettuja komentoja ja työhakemisto (Current Directory) on hakemistoselain.



Tutustu ensin MATLABIN ohjetoimintoihin, sillä ilman niitä et jatkossa tule toimeen. Komentoja ja funktioita on niin paljon, että on mahdotonta muistaa niiden kaikkien syntaksia ulkoa. Ohjeen käyttöohje tulee näkyviin komentoikkunaan kirjoittamalla kehotteeseen (lihavoidut tekstit ovat käyttäjän kirjoittamia komentoja, päätä komennot Enterillä).

```
>> help help
```

Ohjeaiheiden listaus tulee komentoikkunaan sivu kerrallaan näkyviin (lisäsivuja tulee välilyöntinäppäimestä, keskeytys q-näppäimestä) komennoilla

```
>> more on
>> help
```

Aiheiden listauksen alkupää on seuraavan näköinen:

```
HELP topics:
matlab\general      -  General purpose commands.
matlab\ops          -  Operators and special characters.
matlab\lang         -  Programming language constructs.
```

```
matlab\elmat       -  Elementary matrices and matrix manipulation.
matlab\elfun       -  Elementary math functions.
matlab\specfun     -  Specialized math functions.
matlab\matfun      -  Matrix functions - numerical linear algebra.
...
```

Tietystä aihealueesta saa ohjeita klikkaamalla sen linkkiä aiheiden listauksessa tai kirjoittamalla kehotteeseen esimerkiksi

```
>> help matfun

Matrix functions - numerical linear algebra.
  Matrix analysis.
    norm        - Matrix or vector norm.
    normest     - Estimate the matrix 2-norm.
    rank        - Matrix rank.
    det         - Determinant.
    trace       - Sum of diagonal elements.
...
```

Aihealueen tietystä funktiosta saa edelleen ohjeita klikkaamalla sen linkkiä tai seuraavasti

```
>> help trace

 TRACE  Sum of diagonal elements.
    TRACE(A) is the sum of the diagonal elements of A, which is
    also the sum of the eigenvalues of A.
...
```

Komentoikkunassa voi suorittaa sanahakuja `lookfor`-funktiolla. Kun halutaan löytää ohjeista esimerkiksi sana `Euler`, kirjoitetaan

```
>> lookfor Euler
RIGIDODE  Euler equations of a rigid body without external forces.
EULER Simulink 1.x EULER integration algorithm.
```

MATLABin graafinen ohjeselain-ikkuna avautuu komennolla

```
>> helpbrowser
```

tai valitsemalla työpöydän valikosta Help > MATLAB Help. Kun kehotteeseen annetaan komento

```
>> helpwin
```

näyttää MATLAB ohje-selaimessa kaikista funktioista listan, jossa on linkit funktioiden ohjeriveille. Funktio doc näyttää argumenttina annetun funktion referenssisivun ohje-selaimessa. Esimerkiksi

```
>> doc atan2
```

näyttää atan2-valmisfunktion referenssisivun. Demonstraatioikkuna avautuu komennolla

```
>> demo
```

ja siinä on nähtävissä MATLABin ja siihen liittyvien lisäohjelmien käyttöesimerkkejä. Demonstraatioikkuna avautuu myös valitsemalla työpöydän valikosta Help > Demos.

2 MATLABin käyttö laskimena

Aritmeettiset perusoperaatiot ovat `+`, `-`, `*`, `/` ja `^`, joiden lisäksi voidaan käyttää sulkeita `()`. Operaatioiden suoritusjärjestys on tavanomainen: 1. sulkeet `()`, 2. potenssiin korotus `^`, 3. kerto- ja jakolaskut `*` ja `/` vasemmalta oikealle, 4. yhteen- ja vähennyslaskut `+` ja – vasemmalta oikealle. Kokeile seuraavia aritmeettisia operaatioita.

```
>> format compact
>> 4-3*5/8^2
ans =
    3.7656
>> 4-3*(5/8)^2
ans =
    2.8281
>> 5+ans-3*ans^ans
ans =
  -48.9284
```

Viimeisin tulos tallennetaan muuttujaan `ans`, kuten monissa laskimissakin. Komentoriviä voi editoida normaalisti ennen Enterin painamista ja aikaisempia komentoriville kirjoitettuja lausekkeita voi kopioida ja liittää uudelle komentoriville. Edellä kävi ilmi, että MATLAB tunnistaa (tietenkin) kokonaisluvut ja desimaaliluvut, mutta niiden lisäksi myös kompleksiluvut (imaginääriyksikkö on `i` tai `j`) sekä 'luvut' `Inf` (ääretön) ja `NaN` (Not a Number), kuten seuraavasta ilmenee.

```
>> (3-2i)/(4+5i)^3
ans =
  -0.0136 + 0.0018i
>> 6/0
Warning: Divide by zero.
ans =
   Inf
>> 0/0
Warning: Divide by zero.
ans =
   NaN
```

Itseisarvoltaan hyvin suurille tai pienille luvuille on käytettävissä e-notaatio.

```
>> 1.345^45.23
ans =
  6.6391e+005
>> -3.98/4.89e16
ans =
 -8.1391e-017
```

MATLAB suorittaa kaikki aritmeettiset operaatiot kaksinkertaisella tarkkuudella (n. 15 merkitsevää numeroa). Käyttäjä voi kuitenkin valita komentoikkunassa käytettävän tulostusformaatin tietyistä vaihtoehdoista, jotka selviävät antamalla komento

```
>> help format
```

Edellä jo käytettiin komentoa `format compact`, joka jättää tulostuksesta ylimääräiset rivinsiirrot pois, jolloin komentoikkunaan mahtuu kerralla enemmän operaatioita. Komentoikkuna tyhjennetään komennolla `clc` tai työpöydän valikosta Edit > Clear Command Window, jolloin muuttujia (ks. kohta 3) ei kuitenkaan poisteta muuttuja-avaruudesta.

Komentoikkunaan kirjoitettu tai tulostunut teksti ei tallennu automaattisesti. Komentoikkunan sisältö voidaan tallentaa kokonaan tai osittain tekstitiedostoon komennolla `diary`. Kun annetaan komento

```
>> diary tiedostonnimi
```

MATLAB avaa tekstitiedoston *`tiedostonnimi`* ja nauhoittaa siihen kaiken komentoikkunaan tiedoston avaamisen jälkeen tulevan tekstin. Käytettävä *`tiedostonnimi`* voi olla mikä tahansa luvallinen tiedoston nimi, paitsi *`on`* tai *`off`*. Komennolla `diary off` nauhoitus keskeytyy ja komennolla `diary on` se käynnistyy uudelleen. Aikaisemmin `diary`-komennolla tallennettu tekstitiedosto voidaan avata MATLABin editori/debuggeriin työhakemisto-ikkunasta tai komentoriviltä komennolla `uiopen` *`tiedostonnimi`*.

3 Muuttujien käyttö

Edellä nähtiin, että MATLAB käyttää muuttujaa `ans` viimeisimmän tuloksen tallentamiseen ja tätä muuttujaa voidaan käyttää seuraavissa operaatioissa. Käyttäjä voi määritellä itsekin muuttujia sekä tallentaa niihin numeerista dataa ja käyttää niitä jatkossa määrittelemissään lausekkeissa.

```
>> x=12-3^2/26
x =
   11.6538
```

Edellä on määritelty muuttuja `x` ja siihen on sijoitettu lauseke `12-3^2/26`. Merkki = on siis sijoitusoperaattori, jonka avulla muuttujaan sijoitetaan arvo. Muuttuja `x` tulee näkyviin muuttuja-avaruus ikkunan ja sitä voidaan käyttää jatkossa komentoikkunassa operoitaessa. Sijoitusoperaattorin oikealla puolella olevissa lausekkeissa voi esiintyä vain muuttujia, joihin on sijoitettu jokin arvo. Seuraavassa on käytetty muuttujaa `x` uuden muuttujan `y` määrittelemisessä ja määritelty edelleen muuttujat `z` ja `u`. Huomaa, että komentoriville voi kirjoittaa useampia komentoja kerralla pilkulla erotettuina ja ne suoritetaan yhdellä kerralla Enterin painamisen jälkeen.

```
>> y=x-8.97/x, z=2*y+0.2^y, u=x+y-z
y =
   10.8841
z =
   21.7683
u =
    0.7697
```

MATLAB näyttää komentoikkunassa annettujen komentojen aiheuttamat tulosteet automaattisesti, ellei sitä estetä kirjoittamalla puolipiste (;) komennon/komentojen perään.

```
>> w=z-6.3/u;
>> a=x+y; b=x-y;
>>
```

Muuttujien nimien täytyy alkaa kirjaimella, minkä jälkeen voi olla haluttu määrä kirjaimia, numeroita ja alaviivoja, joista MATLAB käyttää 31 ensimmäistä merkkiä. Luvallisia nimiä ovat siis mm.

```
OoLaLaa, Pentti, A4, X_files, a12g67
```

Kiellettyjä muuttujan nimiä ovat sen sijaan esimerkiksi

```
Oo-La-Laa, 4You, %liike, +M, 'masa'
```

Järkevää on luonnollisesti käyttää mahdollisimman kuvaavia muuttujan nimiä. Tiettyjen varattujen muuttujien nimien käyttöä kannattaa välttää, jotta niitä ei määriteltäisi epähuomiossa uudelleen. Tällaisia ovat esimerkiksi

```
>> pi, eps, Inf, NaN
ans =
    3.1416
ans =
  2.2204e-016
ans =
   Inf
ans =
   NaN
```

Kompleksiluvuilla laskettaessa on syytä välttää muuttujien `i` ja `j` käyttöä, koska ne on molemmat varattu imaginääriyksiköksi ja uudelleen määrittely voi aiheuttaa sekaannuksia, kuten seuraavasta esimerkistä käy ilmi.

```
>> i, j
ans =
        0 + 1.0000i
ans =
        0 + 1.0000i
>> j=8, k=i+j
j =
     8
k =
   8.0000 + 1.0000i
```

Istunnossa määritellyt muuttujat nähdään muuttuja-avaruus ikkunasta, mutta ne voi listata myös komentoikkunaan komennolla `who` tai `whos` (kokeile). Muuttujia voi poistaa muuttuja-avaruus ikkunassa tai komennolla `clear` *`muuttujanimi`* (kokeile). Komento `clear all` tai valinta Edit > Clear Workspace työpöydän valikosta poistaa kaikki muuttujat, käytä siis niitä harkiten.

Muuttujat voi tallentaa `mat`-tiedostoon tulevissa istunnoissa käytettäviksi. Tallennus tapahtuu työpöydän valikosta File > Save Workspace As tai komentoikkunasta komennolla `save` *`tiedostonimi`* (tarkennin oltava `mat`). `mat`-tiedostossa olevien muuttujien lataaminen muuttuja-avaruuteen tapahtuu työpöydän valikosta File > Import Data… tai komentoikkunasta komennolla `load` *`tiedostonimi`*. Jos tuotava muuttuja on samanniminen komentoikkunassa olemassa olevan muuttujan kanssa, korvaa se olemassa olevan muuttujan. Kokeile muuttujien tallentamista ja lataamista!

4 Perusfunktiot

MATLABissa on käytettävissä kaikki tavanomaiset matemaattiset perusfunktiot. Kirjoittamalla komentoriville

```
>> help elfun
```

saadaan komentoikkunaan lista kaikista funktioista ja komennolla

```
>> help funktionnimi
```

tulee komentoikkunaan funktion *`funktionnimi`* käyttöohjeet. Seuraavassa on muutamia perusfunktioiden käyttöesimerkkejä.

```
>> x=6*cos(pi/6), y=6*sin(pi/6)
x =
    5.1962
y =
    3.0000
>> z=abs(log(0.5))+floor(atan(1)/sqrt(x))
z =
    0.6931
>> komp=conj(1/(sqrt(z-3)+2))
komp =
   0.3171 + 0.2408i
```

Trigonometristen funktioiden argumenttien tulee olla radiaaneissa. Kokeile perusfunktioiden käyttöä myös omilla esimerkeillä.

5 Vektorilaskentaa

Vaakavektori annetaan kirjoittamalla sen alkiot hakasulkeisiin välilyönneillä erotettuina.

```
>> v0=[2 6 -3 3]
v0 =
     2     6    -3     3
>> v1=[sqrt(2) exp(1) -sin(pi/4) log(2.3)]
v1 =
    1.4142    2.7183   -0.7071    0.8329
```

Erottimena voi olla myöskin pilkku (`,`), mutta tässä käytetään välilyöntiä. Välilyöntejä voi olla erottimessa useampiakin peräkkäin, mutta luvun ja sen etumerkin väliin ei voi laittaa välilyöntiä merkitystä muuttamatta, kuten seuraavasta esimerkistä näkyy.

```
>> v2=[1 -3 2 + 3 -4]
v2 =
     1    -3     5    -4
>> v3=[1 -3 2+ 3 -4]
v3 =
     1    -3     5    -4
>> v4=[1 -3 2 +3 -4]
v4 =
     1    -3     2     3    -4
```

Funktio `length` palauttaa argumenttina annetun vektorin alkioiden lukumäärän (= vektorin pituus).

```
>> L0=length(v0), L4=length(v4)
L0 =
     4
L4 =
     5
```

Samanpituisille vektoreille on määritelty yhteenlasku (+) ja vähennyslasku (`-`), jotka tapahtuvat tavanomaisesti alkioittain. Vektorin kertominen skalaarilla (*) vasemmalta ja oikealta ovat myös määriteltyjä ja tapahtuvat alkioittain. Seuraavassa on esimerkkejä näistä operaatioista.

```
>> sum=v0+v1, ero=v0-v1
sum =
    3.4142    8.7183   -3.7071    3.8329
ero =
    0.5858    3.2817   -2.2929    2.1671

>> tulo1=3*v1, tulo2=v4*pi
tulo1 =
    4.2426    8.1548   -2.1213    2.4987
tulo2 =
    3.1416   -9.4248    6.2832    9.4248  -12.5664
>> virhe=v0+v4
??? Error using ==> +
Matrix dimensions must agree.
```

Vaakavektorin ja skalaarin voi laskea yhteen (tai vähentää), jolloin kyseinen skalaari lisätään (vähennetään) vektorin jokaiseen alkioon erikseen. Tästä toiminnosta käytetään nimitystä skalaarin laajennus, koska operointi tapahtuu laajentamalla skalaari sopivan dimension omaavaksi vektoriksi, jonka kaikki alkiot ovat kyseisen skalaarin suuruisia.

```
>> v5=v0+2, v6=-4-v0
v5 =
     4     8    -1     5
v6 =
    -6   -10    -1    -7
```

Olemassa olevista vektoreista voidaan rakentaa uusia vektoreita esimerkiksi seuraavasti.

```
>> v_uusi=[-0.6*v0 v1-3*v2 v4]
v_uusi =
  Columns 1 through 7
   -1.2000   -3.6000    1.8000   -1.8000   -1.5858   11.7183  -15.7071
  Columns 8 through 13
   12.8329    1.0000   -3.0000    2.0000    3.0000   -4.0000
```

Vektorin alkio poimitaan tai sitä muutetaan seuraavaan tapaan.

```
>> v3(2)=10, poim=v4(5), v4(1)=v3(3)
v3 =
     1    10     5    -4
poim =
    -4
v4 =
     5    -3     2     3    -4
```

Pystyvektori annetaan kirjoittamalla sen alkiot hakasulkeisiin puolipisteillä erotettuina.

```
>> p1=[1; -3; 2]
p1 =
     1
    -3
     2
```

Puolipisteen jälkeen voi olla välilyöntejä ja sen sijasta erottimena voi olla myös rivinsiirto.

```
>> p2=[sqrt(6);  floor(4.567);      ceil(4.567) ]
p2 =
    2.4495
    4.0000
    5.0000

>> p3=[sign(-3.876)
log(2)
log10(2)]
p3 =
   -1.0000
    0.6931
    0.3010
```

Samanpituisille pystyvektoreille on myös määritelty yhteenlasku (+), vähennyslasku (-) ja skalaarilla kertominen (*) alkioittain kuten vaakavektoreillekin. Myös pystyvektorin ja skalaarin voi laskea yhteen tai vähentää ja toiminta on samanlainen kuin vaakavektoreilla.

```
>> p4=2*p1-p2*pi+length(p2)*p3
p4 =
   -8.6953
  -16.4869
  -10.8049
```

```
>> p5=p1-2+p2+3
p5 =
    4.4495
    2.0000
    8.0000
```

Vektori voidaan transponoida operaattorilla `’`, jonka tuloksena vaakavektorista saadaan pystyvektori ja pystyvektorista vaakavektori.

```
>> p_1=v1’, v_1=p1’
p_1 =
    1.4142
    2.7183
   -0.7071
    0.8329
v_1 =
     1    -3     2
```

Operaattori `’` on konjugaatin transpoosi. Jos kompleksiluvuilla halutaan jättää konjugointi pois, voidaan käyttää operaattoria `.’`. Operaattoreiden `’` ja `.’` ero selviää seuraavasta esimerkistä.

```
>> cv1=[1-3i 3+4i -4+3i], cv1_1=cv1’, cv1_2=cv1.’
cv1 =
   1.0000 - 3.0000i   3.0000 + 4.0000i  -4.0000 + 3.0000i
cv1_1 =
   1.0000 + 3.0000i
   3.0000 - 4.0000i
  -4.0000 - 3.0000i
cv1_2 =
   1.0000 - 3.0000i
   3.0000 + 4.0000i
  -4.0000 + 3.0000i
```

Vektoreiden määrittelyyn ja käsittelyyn MATLABissa on myös operaattori `:`. Sen käyttö vaakavektoreiden määrittelyyn selviää seuraavasta esimerkistä. Huomaa, että vektori `row5` on tyhjä (dimensiot 1×0). Tyhjien taulukoiden käyttö on MATLABissa mahdollista (jotkut dimensiot nollia).

```
>> row1=3:7
row1 =
     3     4     5     6     7
```

```
>> row2=2:3:12
row2 =
     2     5     8    11
```

```
>> row3=-6:-4
row3 =
    -6    -5    -4
```

```
>> row4=-3.16:-1.89:-3*pi
row4 =
  -3.1600   -5.0500   -6.9400   -8.8300
```

```
>> row5=1:-2:5
row5 =
   Empty matrix: 1-by-0
```

Operaattorin `:` avulla pystytään myös poimimaan ja muuttamaan vektorin osia. Seuraavassa esimerkissä on muodostettu vaakavektori `v7` ja käytetty `:`-operaattoria siihen. Mieti huolella kussakin vaiheessa suoritetut toiminnot ja kokeile vielä, mikä on `v7(:)`.

```
>> v7=[v0 v4]
v7 =
     2     6    -3     3     5    -3     2     3    -4
>> v7(3:6)
ans =
    -3     3     5    -3
>> v7(2:3:8)
ans =
     6     5     3
>> v7(9:-3:1)
ans =
    -4    -3    -3
>> v7(5:7)=0
v7 =
     2     6    -3     3     0     0     0     3    -4
>> v7(5:2:7)=[2 4]
v7 =
     2     6    -3     3     2     0     4     3    -4
```

Pystyvektoreita voidaan samaan tapaan käsitellä `:`-operaattorilla. Seuraavassa esimerkissä on muodostettu pystyvektori `p6` ja käytetty `:`-operaattoria sen yhteydessä alkioiden poimintaan ja muuttamiseen.

```
>> p6=[p1; p2]
p6 =
    1.0000
   -3.0000
    2.0000
    2.4495
    4.0000
    5.0000
```

```
>> p6(2:2:5)
ans =
   -3.0000
    2.4495
```

```
>> p6(6:-2:1)
ans =
    5.0000
    2.4495
   -3.0000
```

```
>> p6(:)
ans =
    1.0000
   -3.0000
    2.0000
    2.4495
    4.0000
    5.0000
```

```
>> p6(1:3:6)=0
p6 =
     0
    -3
     2
     0
     4
     5
```

```
>> p6(2:2:6)=[10; pi; -7.7]
p6 =
         0
   10.0000
    2.0000
    3.1416
    4.0000
   -7.7000
```

6 Matriisilaskentaa

Matriisi annetaan hakasuluissa vaakariveittäin, vaakarivit erotetaan puolipisteellä tai rivinsiirrolla. Vaakarivin alkiot erotetaan toisistaan välilyönnillä tai pilkulla.

```
>> A=[2 1; 3 -1; -2 6]
A =
     2     1
     3    -1
    -2     6
```

```
>> B=[4 -7 2
5 9 -1]
B =
     4    -7     2
     5     9    -1
```

Matriisin määrittelyssä voidaan käyttää hyväksi myös `:`-operaattoria.

```
>> C=[2:3:8; 3:-2:-1; 1.5:2.4:7]
C =
    2.0000    5.0000    8.0000
    3.0000    1.0000   -1.0000
    1.5000    3.9000    6.3000
>> D=[sqrt(2) sqrt(3) sqrt(4); 4.6:2.1:9; -1 3:4]'
D =
    1.4142    4.6000   -1.0000
    1.7321    6.7000    3.0000
    2.0000    8.8000    4.0000
```

On selvää, että vaaka- ja pystyvektorit ovat matriisien erityistapauksia. Funktio `size` palauttaa argumenttina annetun matriisin dimensiot rivivektorina. Seuraavassa on esimerkkejä `size`-funktion käyttötavoista.

```
>> size(A), size(C)
ans =
     3     2
ans =
     3     3
```

```
>> [r s]=size(B), rs=size(D)
r =
     2
s =
     3
rs =
     3     3
```

Matriisin konjugaatin transpoosi saadaan operaattorilla `'` ja transpoosi ilman konjugointia operaattorilla `.'`. Reaalisella matriisilla ei näillä operaatioilla luonnollisesti ole eroa.

```
>> A
A =
     2     1
     3    -1
    -2     6
```

```
>> A'
ans =
     2     3    -2
     1    -1     6
```

```
>> A.'
ans =
     2     3    -2
     1    -1     6
```

Tiettyjä erityismatriiseja voidaan luoda suoraan valmisfunktioilla. Funktio `ones` luo halutut dimensiot omaavan matriisin, jonka kaikki alkiot ovat ykkösiä ja funktio `zeros` matriisin, jonka kaikki alkiot ovat nollia. Yksikkömatriisi luodaan funktiolla `eye` ja lävistäjämatriisi funktiolla `diag`.

```
>> E=ones(3,5)
E =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
>> F=zeros(3,6)
F =
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
>> G=eye(3)
G =
     1     0     0
     0     1     0
     0     0     1
```

```
>> H=diag([2 -3 8])
H =
     2     0     0
     0    -3     0
     0     0     8
```

Kun `diag`-funktiolle annetaan argumentiksi matriisi, se palauttaa pystyvektorina annetun matriisin päälävistäjän (alkiot joiden rivi- ja sarakenumero on sama).

```
>> diag(A), diag(B)
ans =
     2
    -1
ans =
     4
     9
```

```
>> diag(C)
ans =
    2.0000
    1.0000
    6.3000
```

Lisää matriiseja voidaan myös rakentaa aikaisemmin määriteltyjen matriisien avulla seuraavan esimerkin tapaan.

```
>> K=[A, C; D, [1; 4; 7], ones(3,1)]
K =
    2.0000    1.0000    2.0000    5.0000    8.0000
    3.0000   -1.0000    3.0000    1.0000   -1.0000
   -2.0000    6.0000    1.5000    3.9000    6.3000
    1.4142    4.6000   -1.0000    1.0000    1.0000
    1.7321    6.7000    3.0000    4.0000    1.0000
    2.0000    8.8000    4.0000    7.0000    1.0000
```

Matriisin alkio poimitaan tai sitä muutetaan samalla periaatteella kuin vektoreilla.

```
>> A(2,1), A(3,2)=-10
ans =
     3
A =
     2     1
     3    -1
    -2   -10
```

Operaattorin `:` avulla pystytään poimimaan ja muuttamaan myös matriisin osia. Seuraavassa esimerkissä on sovellettu `:`-operaattoria matriisiin K.

```
>> K(:,2)
ans =
    1.0000
   -1.0000
    6.0000
    4.6000
    6.7000
    8.8000
```

```
>> K(:,1:2:5)
ans =
    2.0000    2.0000    8.0000
    3.0000    3.0000   -1.0000
   -2.0000    1.5000    6.3000
    1.4142   -1.0000    1.0000
    1.7321    3.0000    1.0000
    2.0000    4.0000    1.0000
```

```
>> K(5,:)
ans =
    1.7321    6.7000    3.0000    4.0000    1.0000
>> K(4:5,2:3)
ans =
    4.6000   -1.0000
    6.7000    3.0000

>> K(3:5,2:2:5)=[111 222; 333 444; 555 666]
K =
    2.0000    1.0000    2.0000    5.0000    8.0000
    3.0000   -1.0000    3.0000    1.0000   -1.0000
   -2.0000  111.0000    1.5000  222.0000    6.3000
    1.4142  333.0000   -1.0000  444.0000    1.0000
    1.7321  555.0000    3.0000  666.0000    1.0000
    2.0000    8.8000    4.0000    7.0000    1.0000
```

Samat dimensiot omaavat matriisit voidaan laskea yhteen operaattorilla + ja vähentää operaattorilla -. Matriisin kertominen alkioittain skalaarilla vasemmalta tai oikealta tapahtuu operaattorilla *. Matriisin ja skalaarin voi laskea yhteen (tai vähentää), jolloin kyseinen skalaari lisätään (vähennetään) matriisin jokaiseen alkioon erikseen. Skalaarin laajennus toimii siis matriiseilla samalla tavalla kuin vektoreilla.

```
>> C+2*D
ans =
    4.8284   14.2000    6.0000
    6.4641   14.4000    5.0000
    5.5000   21.5000   14.3000

>> -C-D*3
ans =
   -6.2426  -18.8000   -5.0000
   -8.1962  -21.1000   -8.0000
   -7.5000  -30.3000  -18.3000

>> C+2
ans =
    4.0000    7.0000   10.0000
    5.0000    3.0000    1.0000
    3.5000    5.9000    8.3000

>> 4-D
ans =
    2.5858   -0.6000    5.0000
    2.2679   -2.7000    1.0000
    2.0000   -4.8000         0
```

Tavanomainen matriisien kertolasku suoritetaan operaattorilla *. Vektorin ja matriisin välinen kertolasku on luonnollisesti vain tämän erityistapaus.

```
>> L=A*B
L =
    13    -5     3
     7   -30     7
   -58   -76     6

>> B*p1
ans =
    29
   -24

>> L1=A*C
??? Error using ==> *
Inner matrix dimensions must agree.

>> C*p2
ans =
   64.8990
    6.3485
   50.7742
```

Neliömatriisin A korottaminen potenssiin n (n positiivinen kokonaisluku) tarkoittaa sellaista matriisituloa, jossa on n kpl tekijöitä A. Myös matriisin korottaminen mielivaltaiseen reaali- tai kompleksilukupotenssiin on MATLABissa määritelty, määritelmä perustuu matriisin ominaisarvoihin. Matriisin potenssiin korotus operaattori on ^. Seuraavassa on esimerkki matriisin C kokonaislukupotenssista ja kompleksilukupotenssista (huomaa, että et voi kirjoittaa pii tai (pi)i).

```
>> C^3
ans =
  292.7000  440.6600  588.6200
   76.3500  114.7300  153.1100
  227.8950  343.1010  458.3070

>> C^(1.29+pi*i)
ans =
   4.3194 + 4.4441i   6.5012 + 6.6890i   8.6831 + 8.9340i
   1.1247 + 1.1573i   1.6929 + 1.7418i   2.2611 + 2.3263i
   3.3631 + 3.4602i   5.0619 + 5.2081i   6.7607 + 6.9560i
```

MATLABissa on käytettävissä monia tavanomaisia matriisilaskennan operaatioita valmisfunktioina. Listan matriisifunktioista saa komentoikkunaan kirjoittamalla komentoriville pyynnön

```
>> help matfun
```

Esimerkiksi funktio `det` laskee neliömatriisin determinantin, `inv` käänteismatriisin ja `eig` ominaisarvot ja –vektorit.

```
>> d=det(D)
d =
   -5.5461
>> D_1=inv(D)
D_1 =
   -0.0721    4.9044   -3.6963
    0.1674   -1.3806    1.0773
   -0.3321    0.5851   -0.2719
```

```
>> [o_vekt, o_arv]=eig(D)
o_vekt =
   -0.1992   -0.9306    0.9249
   -0.5940    0.1030   -0.3158
   -0.7794    0.3512    0.2119
o_arv =
   11.2175         0         0
         0    1.2823         0
         0         0   -0.3856
```

Lineaarinen yhtälöryhmä $Ax = b$ (A $n \times n$-matriisi, x ja b $n \times 1$-vektoreita) voidaan luonnollisesti ratkaista kääntämällä matriisi A, eli $x = A^{-1}b$, mutta MATLABissa on käytettävissä tähän tarkoitukseen tehokkaampi operaattori `\`, joka ratkaisee yhtälöryhmän eliminointikeinolla. Operaattorilla `\` voidaan lisäksi etsiä yli- ja alimääriteltyjen (A ei ole neliömatriisi) yhtälöryhmien pienimmän neliösumman ratkaisu.

```
>> D,p1
D =
    1.4142    4.6000   -1.0000
    1.7321    6.7000    3.0000
    2.0000    8.8000    4.0000
p1 =
     1
    -3
     2
```

```
>> x=inv(D)*p1
x =
  -22.1778
    6.4637
   -2.6312
>> x=D\p1
x =
  -22.1778
    6.4637
   -2.6312
```

Lineaarinen yhtälöryhmä $xA = b$ (A $n \times n$-matriisi, x ja b $1 \times n$-vektoreita) voidaan ratkaista kääntämällä matriisi A, eli $x = bA^{-1}$, mutta tähänkin tarkoitukseen on käytettävissä tehokkaampi operaattori `/`, jolla voidaan lisäksi etsiä yli- ja alimääriteltyjen (A ei ole neliömatriisi) yhtälöryhmien pienimmän neliösumman ratkaisu.

```
>> x=p1'*inv(D)
x =
   -1.2385   10.2163   -7.4719
```

```
>> x=p1'/D
x =
   -1.2385   10.2163   -7.4719
```

7 Taulukkolaskentaa

Taulukkolaskentaan on käytettävissä alkioittain toimivat operaattorit `+`, `-`, `.'`, `.*`, `./`, `.\` ja `.^`. Näistä `+`, `-`, `.'` olivat esillä jo matriisilaskennan yhteydessä, koska ne toimivat alkioittain sekä matriisilaskennassa ja taulukkolaskennassa. Taulukkolaskennan operaatiot ovat määriteltyjä samat vastindimensiot omaavien taulukoiden välillä, poikkeuksen tästä muodostavat kuitenkin taulukon ja skalaarin väliset operaatiot, jotka suoritetaan edellä kuvattua skalaarin laajennusta käyttäen.

Operaattori `.*` on kertolasku alkioittain. Operaation tuloksena on taulukko, jonka alkiot ovat operaatioon osallistuvien taulukoiden vastinalkioiden tulot. Huom! Funktio `rand` antaa satunnaislukuja.

```
>> T1=ceil(10*rand(3,4)-5.5)
T1 =
     3    -2     2     1
     2    -2    -2    -1
    -2     0     3     2
>> T1.*T2
ans =
     3    -2     0     4
     0    -6    -6     2
    -4     0    -6    -4
```

```
>> T2=floor(8*rand(3,4)-3.3)
T2 =
     1     1     0     4
     0     3     3    -2
     2     4    -2    -2
>> T2.*T1
ans =
     3    -2     0     4
     0    -6    -6     2
    -4     0    -6    -4
```

Operaattori `./` on oikeanpuoleinen jakolasku alkioittain. Operaation tuloksena on taulukko, jonka alkiot saadaan jakamalla operaattorin vasemmalla puolella olevan taulukon alkiot sen oikealla puolella olevan taulukon vastinalkioilla.

```
>> T1./T2
Warning: Divide by zero.
ans =
    3.0000   -2.0000       Inf    0.2500
       Inf   -0.6667   -0.6667    0.5000
   -1.0000         0   -1.5000   -1.0000
>> T2./T1
Warning: Divide by zero.
ans =
    0.3333   -0.5000         0    4.0000
         0   -1.5000   -1.5000    2.0000
   -1.0000      -Inf   -0.6667   -1.0000
```

Operaattori `.\` on vasemmanpuoleinen jakolasku alkioittain. Operaation tuloksena on taulukko, jonka alkiot saadaan jakamalla operaattorin oikealla puolella olevan taulukon alkiot sen vasemmalla puolella olevan taulukon vastinalkioilla.

```
>> T1.\T2
Warning: Divide by zero.
ans =
    0.3333   -0.5000         0    4.0000
         0   -1.5000   -1.5000    2.0000
   -1.0000      -Inf   -0.6667   -1.0000
>> T2.\T1
Warning: Divide by zero.
ans =
    3.0000   -2.0000       Inf    0.2500
       Inf   -0.6667   -0.6667    0.5000
   -1.0000         0   -1.5000   -1.0000
```

Operaattori `.^` on potenssiin korotus alkioittain. Operaation tuloksena on taulukko, jonka alkiot saadaan korottamalla operaattorin vasemmalla puolella olevan taulukon alkiot sen oikealla puolella olevan taulukon vastinalkioiden mukaisiin potensseihin.

```
>> T1.^T2
ans =
    3.0000   -2.0000    1.0000    1.0000
    1.0000   -8.0000   -8.0000    1.0000
    4.0000         0    0.1111    0.2500
>> T2.^T1
ans =
    1.0000    1.0000         0    4.0000
         0    0.1111    0.1111   -0.5000
    0.2500    1.0000   -8.0000    4.0000
```

Kohdassa 4 esitellyt matemaattiset perusfunktiot ovat myös luonteeltaan taulukko-operaatioita. Kun perusfunktion argumentiksi annetaan taulukko, operoi perusfunktio siihen alkioittain.

```
>> T3=abs(T2), T4=sin(T1), T5=sqrt(T3).^(1./T4)
T3 =
     1     1     0     4
     0     3     3     2
     2     4     2     2
T4 =
    0.1411   -0.9093    0.9093    0.8415
    0.9093   -0.9093   -0.9093   -0.8415
   -0.9093         0    0.1411    0.9093
T5 =
    1.0000    1.0000         0    2.2790
         0    0.5466    0.5466    0.6624
    0.6831       Inf   11.6567    1.4640
```

8 Merkkijonot

MATLABissa voidaan tallentaa (`char` tietotyyppiseen) muuttujaan merkkijonoja. Merkkijono, jossa on n kpl merkkejä, on $1 \times n$ -taulukko. Merkkijono annetaan yksinkertaisten lainausmerkkien välissä.

```
>> a='Merkurius',b='Venus'
a =
Merkurius
b =
Venus
```

Merkkijonoja voidaan yhdistellä tavanomaisia vaakavektorioperaatioita käyttäen.

```
>> c=[a ' ja ' b]
c =
Merkurius ja Venus
```

Merkkijonon pituus saadaan selville `length`-funktiolla.

```
>> length(c)
ans =
    18
```

Samanpituisia merkkijonoja voidaan tallentaa tavalliseen kaksiulotteiseen taulukkoon. Eripituisia merkkijonoja voidaan tallentaa esimerkiksi ns. solutaulukkoon.

```
>> Planeetat=['Merkurius'; 'Venus    '; 'Tellus   ';
'Mars     '; 'Jupiter  '; 'Saturnus '
'Neptunus '; 'Uranus   '; 'Pluto    ']
Planeetat =
Merkurius
Venus
Tellus
Mars
Jupiter
Saturnus
Neptunus
Uranus
Pluto
>> size(Planeetat)
ans =
     9     9
```

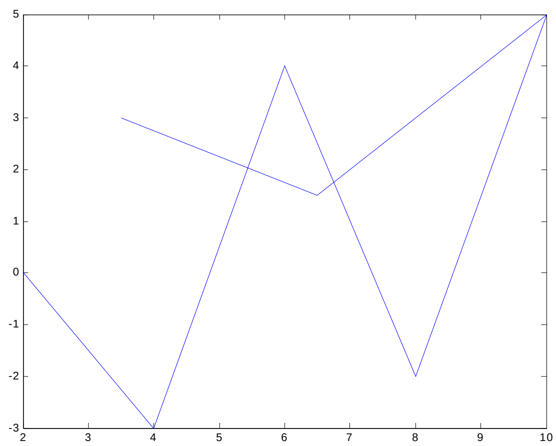
Funktiot `int2str` ja `num2str` muuntavat kokonaisluvun ja reaaliluvun merkkijonoksi ja funktio `str2num` muuntaa merkkijonon vastaavaksi luvuksi.

```
>> N=5; k=2/N^2;
>> ['Muuttujan N arvo on ' int2str(N) ', h= ' num2str(k)]
ans =
Muuttujan N arvo on 5, h= 0.08

>> s='22'; t='33';
>> u=str2num(s)+str2num(t)
u =
    55
```

9 Tasokäyrien piirtäminen

Tasokäyrien peruspiirtämisfunktio on `plot`, jonka yksinkertaisin syntaksi on muotoa `plot(x,y)`, missä x ja y ovat kaksi samanpituista vektoria. Komento `plot(x,y)` piirtää kuvaikkunaan oletusasetuksin pistejoukon, jonka pisteiden koordinaatit muodostuvat vektoreiden x ja y vastinkomponenteista. Piirretyt pisteet yhdistetään suorilla viivoilla komponenttien järjestyksen mukaisesti. Oheisessa kuvassa on piirretty vektori bb vektorin aa 'funktiona'.

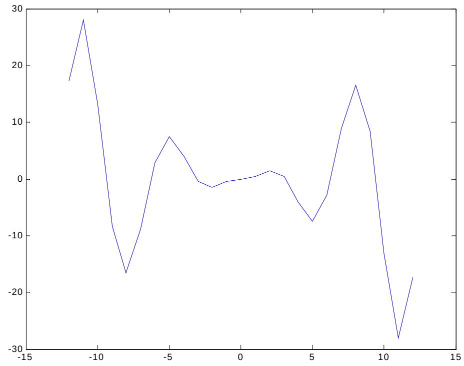



```
>> aa=[2 4 6 8 10 6.5 3.5];
>> bb=[0 -3 4 -2 5 1.5 3];
>> plot(aa,bb)
```

Muodossa $y = f(x),\ x \in [a,b]$ annettujen käyrien piirtäminen perustuu joukkoon argumentin x arvoja ja niitä vastaavaan joukkoon funktion f arvoja, jotka sijoitetaan vektoreihin x ja y. Kun arvoja on riittävästi, näyttää pistejoukoista laadittu murtoviiva sileältä.

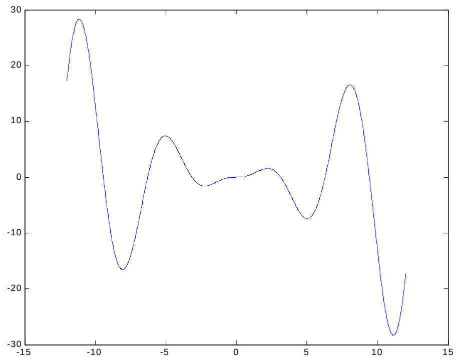
Seuraavassa esimerkissä on käytetty liian vähän argumentin arvoja, jolloin kuvaaja näyttää murtoviivalta.

```
>> x=-12:1:12;
>> y=x.^2.*sin(x)./(1+sqrt(abs(x)));
>> plot(x,y)
```



Lisäämällä argumentin arvojen lukumäärää saadaan aikaan sileältä näyttävä kuvaaja. Uusi kuvaikkuna avataan komennolla `figure`. Komento `figure(k)` (k positiivinen kokonaisluku) avaa tai ottaa käyttöön aikaisemmin avatun kuvaikkunan `k`, johon piirtämiskomennot kohdistuvat, kunnes toisin määrätään.

```
>> x=-12:0.02:12;
>> y=x.^2.*sin(x)./(1+sqrt(abs(x)));
>> figure(2)
>> plot(x,y)
```



Huomaa, että edellä on käytetty taulukko-operaatioita `.^`, `.*` ja `./` funktion arvojen generoinnissa ja muista, että perusfunktiot `sin`, `abs` ja `sqrt` toimivat vektoriargumenteilla alkioittain.

Käyttämällä `plot`-funktiota syntaksilla `plot(x1,y1,jono1,x2,y2,jono2,...)` voidaan piirtää useampia käyriä (vektoriparit `xk,yk`) samaan kuvaikkunaan ja lisäksi vaikuttaa merkkijonoilla `jonok` syntyvien kuvaajien ulkoasuun. Merkkijono `jonok` sisältää tiedot viivan väristä, viivatyypistä ja korostusmerkistä, jotka annetaan oheisen taulukon mukaisilla symboleilla. Symbolien järjestyksellä ei ole merkkijonossa väliä, mutta yleensä ne annetaan järjestyksessä väri, tyyppi, merkki.

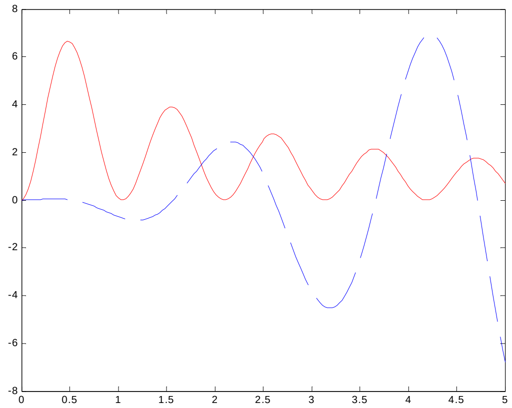
Väri		Viivatyyppi		Korostusmerkki	
y	yellow	-	yhtenäinen viiva	o, +, *, x	merkit o, +, *, x
m	magenta	--	katkoviiva	s	neliö
c	cyan	:	pisteviiva	d	timantti
r	red	-.	pistekatkoviiva	^, >, <, v	kolmioita
g	green	puuttuu	ei viivaa	h, p	monikulmioita
b	blue			puuttuu	ei merkkiä
w	white				
k	black				

Seuraavassa esimerkissä on tehty vektori `x` `linspace`-funktiolla (`linspace(a,b,n)` muodostaa välille `[a,b]` tasajaolla n kpl x arvoja). Sitten on määritelty vastaavat funktioiden `y1` ja `y2` arvot ja avattu kuvaikkuna komennolla `figure(3)`. Kuvaaja `y1` piirretään punaisella yhtenäisellä viivalla ja `y2` sinisellä katkoviivalla. Kuvaajat ovat seuraavalla sivulla ylhäällä oikealla.

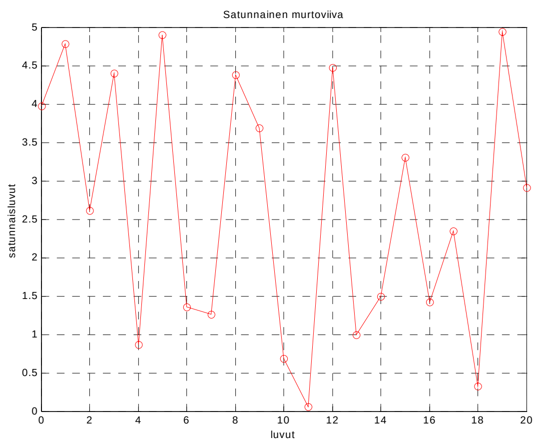
```
>> x=linspace(0,5,200);
>> y1=10./(x+1).*sin(3*x).^2;
>> y2=x.*log(1+x).*cos(3*x);
>> figure(3)
>> plot(x,y1,'r-',x,y2,'b--')
```

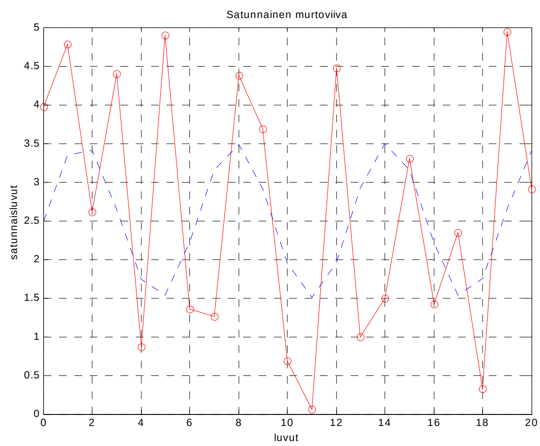
Tarkastellaan seuraavaksi komentosarjaa

```
>> figure(4)
 >> x=0:20;
 >> y=5*rand(1,21);
 >> plot(x,y,’r-o’)
 >> grid on
 >> xlabel(’luvut’)
 >> ylabel(’satunnaisluvut’)
 >> title(’Satunnainen murtoviiva’)
 >> hold on
 >> z=2.5+sin(x);
 >> plot(x,z,’b:’)
```



Aluksi luodaan vektorit x ja y ja sitten piirretään vektori y vektorin x funktiona punaisella yhtenäisellä viivalla käyttäen korostusmerkkinä symbolia o (pieni o-kirjain). Komento `grid on` tuottaa kuvaan katkoviivalla piirretyn hilaruudukon, ruudukko voidaan poistaa komennolla `grid off`. Komennolla `xlabel(’`*`merkkijono1`*`’)` annetaan vaaka-akselin viereen tuleva teksti *`merkkijono1`*, komennolla `ylabel(’`*`merkkijono2`*`’)` annetaan pysty-akselin viereen tuleva teksti *`merkkijono2`* ja komennolla `title(’`*`merkkijono3`*`’)` tulee kuvaan otsikko *`merkkijono3`*. Komento `plot` tyhjentää oletusarvoisesti piirtämisen kohteena olevan kuvaikkunan, mutta sen tyhjennyksen voi estää komennolla `hold on`, jolloin voidaan jatkaa piirtämistä samaan kuvaan. Komennolla `hold off` voidaan ottaa asetettu kuvan pito pois päältä. Aktiivisena oleva kuvaikkuna voidaan tyhjentää manuaalisesti komennolla `clf`. Esimerkissä on vielä luotu vektori z ja piirretty se sinisellä pisteviivalla samaan kuvaan.

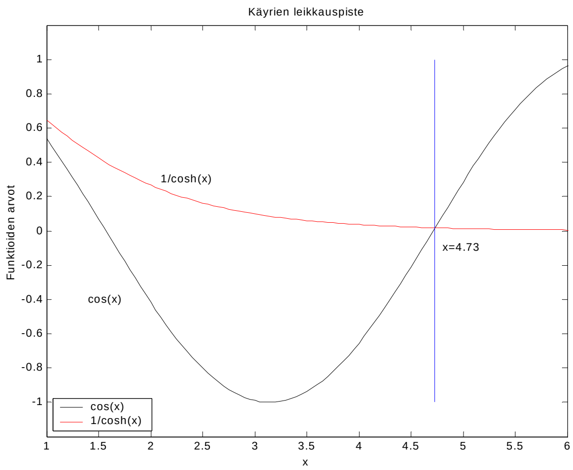




Seuraavassa esimerkissä on esitelty vielä joitakin komentoja, joilla voidaan vaikuttaa mm. syntyvän kuvan ulkoasuun. Huomattakoon aluksi, että vektoreita `cos(x)` ja `1./cosh(x)` ei tarvinnut välttämättä määritellä ennen `plot`-komentoa ja että `plot`-komennon kolmas osa piirtää pystysuoran viivan pisteiden `(4.73,-1)` ja `(4.73,1)` välille. Komennoilla `xlim` ja `ylim` voidaan antaa vaaka- ja pystyakselin arvoalueet ja komennolla `axis` molempien akseleiden arvoalueet.

```
>> figure(5)
>> x=-2:.05:8;
>> plot(x,cos(x),’k’,x,1./cosh(x),’r’,[4.73 4.73],[-1 1],’b’)
>> xlabel('x'); ylabel('Funktioiden arvot');
>> title(’Käyrien leikkauspiste’)
>> xlim([2 7]); ylim([-0.8 0.8])
```

```
>> axis([1 6 -1.2 1.2])
>> legend('cos(x)','1/cosh(x)',3)
>> text(4.8,-.1,'x=4.73')
>> text(2.1,.3,'1/cosh(x)')
>> text(1.4,-.4,'cos(x)')
```



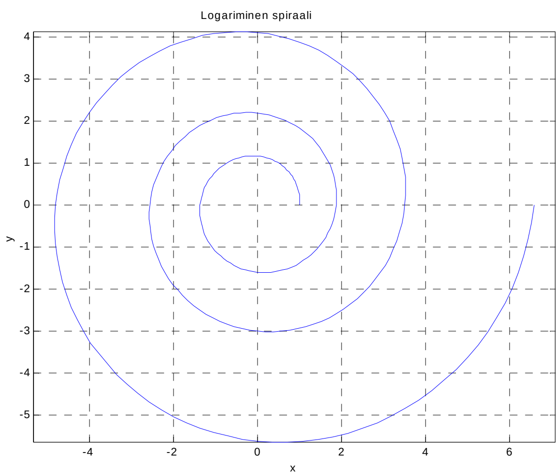
Komennolla `legend` saadaan kuvaan käyrien tunnistetaulu, jossa tunnisteet annetaan käyrien piirtämisjärjestyksessä ja viimeinen parametri määrittelee sen sijainnin (3 = vasen alakulma), Koordinaatistoon voidaan sijoittaa `text`-komennolla merkkijonoina annettavia tekstejä, jolloin aluksi annetaan tekstin vasemman yläkulman koordinaatit.

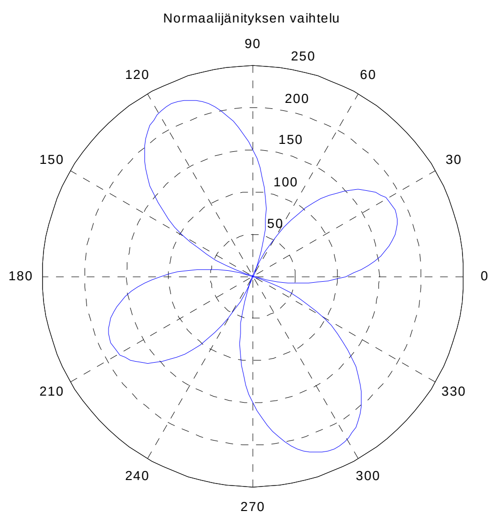
Parametrimuodossa $x = f(t), y = g(t)$ $t \in [a,b]$ annetun käyrän piirtäminen sujuu aivan samalla tavalla kuin edellä esitettiin, nyt vaaka- ja pystyakselille tulevat vektorit `x` ja `y` määritellään vain parametrivektorin `t` avulla. Seuraavassa esimerkissä on piirretty kolme kierrosta logaritmista spiraalia käyttäen hyväksi sen parametriesitystä. Kuvaaja on sivun alareunassa vasemmalla.

```
>> t=0:pi/50:6*pi;
>> x=exp(0.1*t).*cos(t);
>> y=exp(0.1*t).*sin(t);
>> figure(6)
>> plot(x,y)
>> grid on; axis equal
>> title('Logaritminen spiraali')
>> xlabel('x'); ylabel('y')
```

Napakoordinaatistossa käyrä annetaan muodossa $r = f(\theta)$, $\theta \in [\alpha,\beta]$. Käyrän piirto suoritetaan komennolla `polar(θ,r(θ))`. Seuraavassa on esimerkki napakoordinaatiston käyrän piirtämisestä, kuvaaja on sivun alareunassa oikealla.

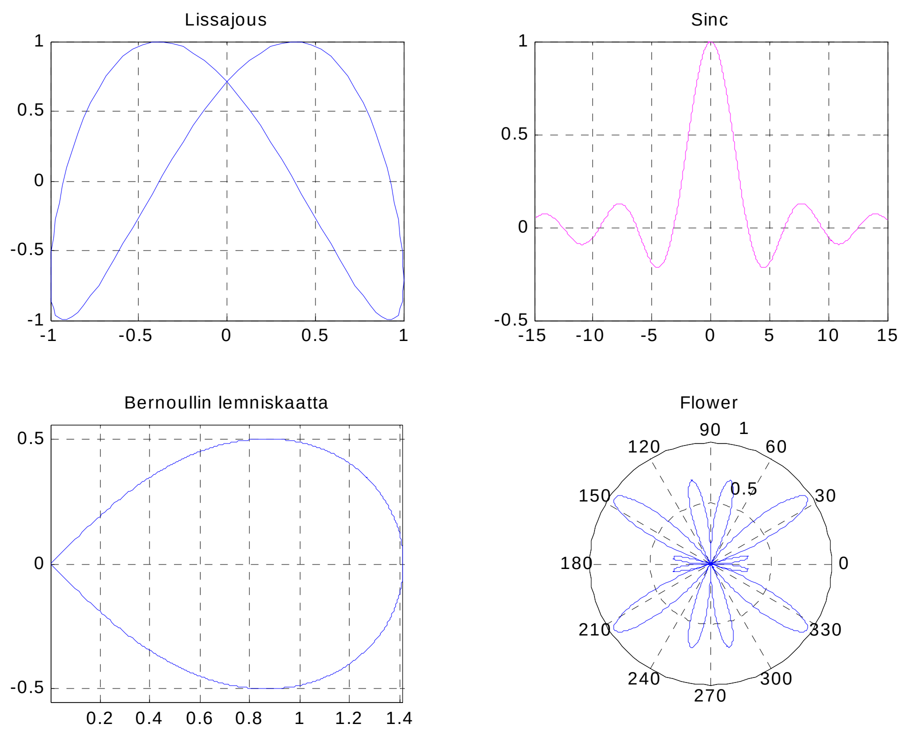
```
>> figure(7)
>> t=0:pi/100:2*pi;
>> polar(t,-20+130*cos(2*t)+160*sin(2*t))
>> title('Normaalijännityksen vaihtelu')
```





Kuvaikkuna voidaan jakaa ali-ikkunoihin komennolla `subplot`, jonka syntaksi on `subplot(m,n,k)` tai `subplot(mnk)`. Näin syntyy ali-ikkunoiden matriisi, jossa on `m` kpl vaakarivejä ja n kpl pystyrivejä. Luku k määrää, monesko ikkuna asetetaan aktiiviseksi, numerointi alkaa vasemmasta yläkulmasta ja etenee vaakariveittäin vasemmalta oikealle Komennolla `subplot` vaihdetaan myös aktiivista ali-ikkunaa. Seuraavassa on esimerkki ali-ikkunoiden käytöstä.

```
>> figure(8)
>> subplot(2,2,1)
>> theta=linspace(0,2*pi,101);
>> plot(sin(theta),sin(2*theta+pi/4))
>> grid on
>> title('Lissajous')
>> subplot(2,2,2)
>> x=linspace(-15,15,1000); y=sin(x)./x;
>> plot(x,y,'m-')
>> xlim([-15 15])
>> grid on
>> title('Sinc')
>> subplot(2,2,3)
>> fii=linspace(-pi/4,pi/4,1000);
>> plot(cos(fii).*sqrt(2*cos(2*fii)),sin(fii).*sqrt(2*cos(2*fii)))
>> grid on
>> axis equal
>> title('Bernoullin lemniskaatta')
>> subplot(2,2,4)
>> psi=linspace(0,2*pi,1000);
>> r=sin(3*psi).*cos(5*psi);
>> polar(psi,r)
>> title('Flower')
```



10 ASCII-tiedostojen I/O

ASCII-muotoista dataa tallennetaan tiedostoon tai kirjoitetaan näytölle funktiota `fprintf` käyttäen. Tiedostoon tallennettaessa se avataan `fopen`-komennolla ja suljetaan kirjoituksen päätyttyä `fclose`-komennolla. Funktio `fprintf` muuntaa dataa merkkijonoiksi ja kirjoittaa ne näytölle tai tallentaa tiedostoon annetun muotoiluformaatin mukaisesti. Syntaksi on seuraavaa muotoa

```
lukum=fprintf(tunniste,format,A,...)
```

Kirjoitettava data sijaitsee matriisissa `A` ja sen jälkeen olevissa muissa matriiseissa (tarkemmin sanoen matriisien reaaliosissa). Palautettava arvo `lukum` (ei pakollinen) on kirjoitettujen tavujen lukumäärä ja `tunniste` on luku, joka viittaa tiedostoon tai on standardi output (=1=näyttö) tai standardi error (=2). Näytölle kirjoitettaessa `tunniste` voi puuttua. `format`-merkkijono sisältää muotoilumäärittelyjä ja halutun valikoiman sellaisenaan tulostuvaa tekstiä ja tulostumattomia merkkejä (esim. rivinsiirto, tabulaattori, jne.). MATLABin `fprintf`-funktio on vektoroitu, ts. se käyttää `format`-merkkijonoa yhä uudelleen niin kauan kuin siihen sopivaa kirjoitettavaa riittää.

Seuraavassa on kaksi esimerkkiä näytölle kirjoittamisesta.

```
>> a=rand(1)*10; b=rand(1)*5;
>> fprintf(1,'1. puoliakseli = %g \n2. puoliakseli = %g \nEllipsin ala
= %g',a,b,pi*a*b)
1. puoliakseli = 9.21813
2. puoliakseli = 3.69104
Ellipsin ala = 106.891
>> x=0:0.1:0.5; y=[x; exp(x)]
y =
         0    0.1000    0.2000    0.3000    0.4000    0.5000
    1.0000    1.1052    1.2214    1.3499    1.4918    1.6487
>> fprintf('x= %4.2f  exp(x)= %10.8f\n',y)
x= 0.00  exp(x)= 1.00000000
x= 0.10  exp(x)= 1.10517092
x= 0.20  exp(x)= 1.22140276
x= 0.30  exp(x)= 1.34985881
x= 0.40  exp(x)= 1.49182470
x= 0.50  exp(x)= 1.64872127
```

Tarkastellaan seuraavaksi datan tallentamista tiedostoon. Tehdään tallennettava data aluksi `rand`-funktion avulla.

```
>> data1=rand(5,3)*200-100
data1 =
  -69.8254   18.7126   63.5949
   39.5797   -0.6895   32.0455
  -24.3254   79.9538  -31.6059
   72.0023   64.3258  -42.0548
   70.7310   28.9821  -31.7613
```

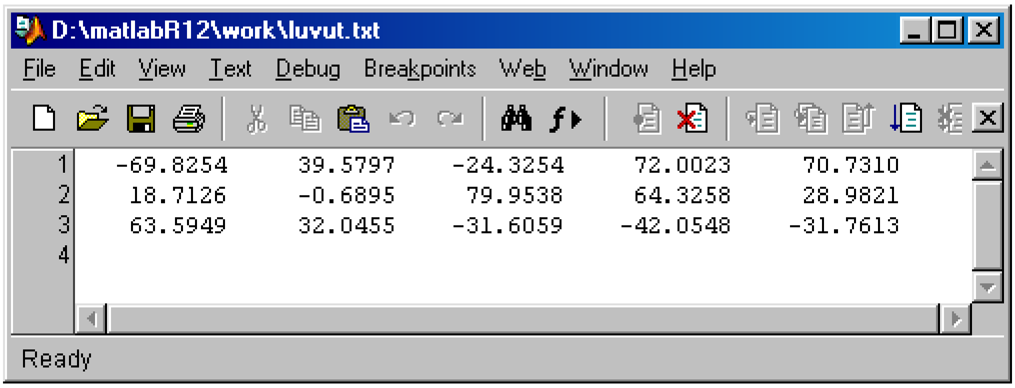
Avataan sitten komennolla `fopen` tiedosto `luvut.txt` kirjoitusoikeuksin (`'w'`). Tiedoston tunnisteeksi `tun` tulee luku 3, jolla jatkossa voidaan siis viitata tiedostoon `luvut.txt`.

```
>> tun=fopen('luvut.txt','w')
tun =
     3
```

Kirjoitetaan seuraavaksi `data1` tiedostoon `luvut.txt` `fprintf`-funktiota käyttäen.

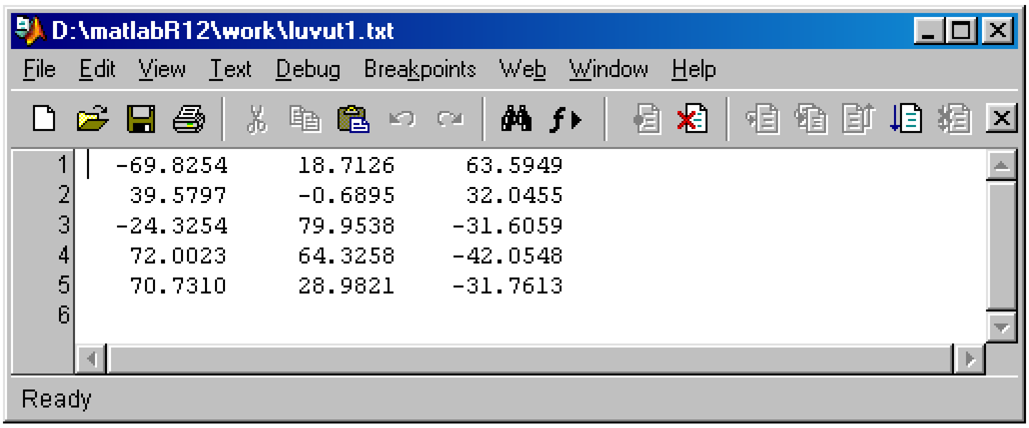
```
>> luk=fprintf(tun,'%10.4f  %10.4f  %10.4f  %10.4f  %10.4f  \n',data1)
luk =
   183
>> fclose(tun)
ans =
     0
```

`fprintf` palauttaa argumentin `luk` arvon `183`, joten tallennettiin 183 tavua. Lopuksi suljetaan tiedosto `luvut.txt` `fclose`-komennolla tiedoston tunnistetta `tun` käyttäen. `fclose` palauttaa arvon `0`, joka on merkki onnistuneesta sulkemisesta. Sulkemisen jälkeen tiedosto `luvut.txt` voidaan avata esimerkiksi MATLABin editori/debuggeriin alla olevan kuvan mukaisesti. Kuvasta näkyy, että `fprintf` tallentaa matriisin sarakkeittain.



Jos matriisin `data1` halutaan tallentuvan tiedostoon `luvut1.txt` vaakariveittäin, voidaan käyttää hyväksi esimerkiksi transponointia seuraavasti.

```
>> tun1=fopen('luvut1.txt','w')
tun1 =
     4
>> luk1=fprintf(tun1,'%10.4f  %10.4f  %10.4f  \n',data1')
luk1 =
   185
>> fclose(tun1)
ans =
     0
```



Funktiolla `fscanf` luetaan tiedostosta ASCII-muotoista dataa. Käytettäessä funktiota `fscanf` muodossa

```
A=fscanf(tunniste,format)
```

se lukee matriisiin `A` kaiken datan tiedostosta, johon `tunniste` viittaa ja muuntaa lukemansa datan tallennusmuotoon, jonka `format`-merkkijono määrittelee. MATLABin `fscanf`-funktio on vektoroitu, ts. se käyttää `format`-merkkijonoa yhä uudelleen niin kauan kuin siihen sopivaa luettavaa riittää.

Seuraavassa esimerkissä luetaan edellä tallennetussa tiedostossa `luvut.txt` oleva data matriisiin `D` `fscanf`-komentoa käyttäen. Ennen lukemista on tiedosto `luvut.txt` avattava lukuoikeuksin (`'r'`) `fopen`-komennolla. Lukemisen jälkeen tiedosto `luvut.txt` suljetaan `fclose`-komennolla sen tunnistetta käyttäen. Nähdään, että tiedosto luettiin riveittäin pystyvektoriksi `D`. Funktiolla `reshape` voidaan palauttaa alkuperäinen matriisin muoto.

```
>> tun=fopen('luvut.txt','r')
tun =
     3
>> D=fscanf(tun, '%g');
>> fclose(tun)
ans =
     0
>> D
D =
  -69.8254
   39.5797
  -24.3254
   72.0023
   70.7310
   18.7126
   -0.6895
   79.9538
   64.3258
   28.9821
   63.5949
   32.0455
  -31.6059
  -42.0548
  -31.7613
>> D1=reshape(D,5,3)
D1 =
  -69.8254   18.7126   63.5949
   39.5797   -0.6895   32.0455
  -24.3254   79.9538  -31.6059
   72.0023   64.3258  -42.0548
   70.7310   28.9821  -31.7613
```

Käytettäessä funktiota `fscanf` muodossa

```
[A,lukum2]=fscanf(tunniste,format,lukum1)
```

se toimii muuten kuten edellä, mutta lukee tiedostosta matriisiin `A` vain niin monta alkiota, jonka `lukum1` määrittelee ja palautettu arvo `lukum2` sisältää onnistuneesti luettujen alkioiden lukumäärän. Jos `lukum1=n` (n kokonaisluku), luetaan n alkiota vektoriin A, jos `lukum1=inf`, luetaan vektoriin A niin monta alkiota kun tiedostossa on ja jos `lukum1=[n,m]`, alkioita luetaan sarakkeittain nxm-matriisiin A.

Luetaan vielä tiedostoa `luvut.txt` `fscanf`-funktion jälkimmäistä syntaksia käyttäen.

```
>> tun=fopen('luvut.txt','r')
tun =
     3
>> [E,luk2]=fscanf(tun,'%g',[5,2])
E =
  -69.8254   18.7126
   39.5797   -0.6895
  -24.3254   79.9538
   72.0023   64.3258
   70.7310   28.9821
luk2 =
    10
```

Tiedostosta `luvut.txt` luettiin siis matriisin E kaksi saraketta eli yhteensä `10` alkiota, joten kaikkia tiedostossa olevaa dataa ei luettu. Selvitetään sitten tiedosto-osoittimen paikka komennolla `ftell`, joka kertoo monenko tavun päässä (`119`) osoitin on tiedoston alussa. Siirretään tiedosto-osoitin komennolla `fseek` paikkaan `0` tavua tiedoston alusta (`'bof'`) lukien, jolloin sitä voidaan jälleen lukea alusta lähtien. Luetaan vielä `fscanf`-funktiolla matriisiin F kaikki tiedostossa oleva data.

```
>> osoitin=ftell(tun)
osoitin =
   119
>> paikka=fseek(tun,0,'bof')
paikka =
     0
>> [F,luk3]=fscanf(tun,'%g',[5,3])
F =
  -69.8254   18.7126   63.5949
   39.5797   -0.6895   32.0455
  -24.3254   79.9538  -31.6059
   72.0023   64.3258  -42.0548
   70.7310   28.9821  -31.7613
luk3 =
    15
```

Selvitetään lopuksi tiedosto-osoittimen paikka lukemisen päätyttyä `ftell`-komennolla ja siirretään `fseek`-komennolla osoitin paikkaan `0` tavua tiedoston lopusta (`'eof'`) lukien.

```
>> osoitin=ftell(tun)
osoitin =
   180
>> paikka=fseek(tun,0,'eof')
paikka =
     0
>> osoitin=ftell(tun)
osoitin =
   183
>> fclose(tun)
```