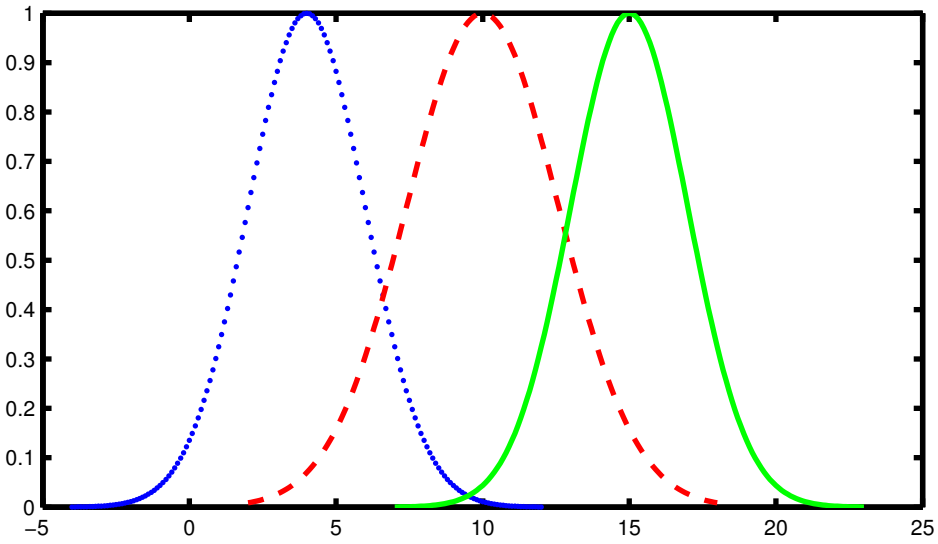
MATLAB OPAS

Matti Pastell
matti.pastell@helsinki.fi
Maataloustieteiden laitos, Helsingin Yliopisto

11. tammikuuta 2010



http://papers.mpastell.com/matlab_opas.pdf

Sisältö

1 Johdanto

Tämän oppaan tarkoitus on opettaa MATLABin[1] peruskäyttö ja hyödyllisiä komentoja. Mukana on myös muutama hieman edistyneempi esimerkki. MATLAB on kaupallinen ohjelma ja sen hankintahinta on melko suuri, tosin opiskelijaversio on huomattavasti halvempi, n. 70 e. Helsingin yliopistolla on ohjelmistoon laaja lisenssi, joka ei kuitenkaan kata ohjelman asennusta opiskelijoiden omille koneille. Kotoa ohjelmaa pääsee käyttämään ottamalla yhteyden yliopiston ruuvi.helsinki.fi (edellyttää Unix lupaa).

Pyrin täydentämään opusta tarpeen mukaan ja viimeisimmän version löytää kotisivuiltani. Suositan aloittelijalle komentojen testaamista samalla, kun luet opasta. Parannusehdotukset ja virheilmoitukset voi lähettää sähköpostitse matti.pastell@helsinki.fi. Opas on kirjoitettu LaTeX- kuvauskielellä.

Usein kysytään eikö olisi helpompaa käyttää Exceliä, SPSS tms. Komentorivipohjaiset ohjelmat saattavat tuntua aluksi vaikeilta käyttää, mutta niiden ominaisuudet ovat ylivoimaisia esim. suurten datamäärien käsittelyssä, omien funktioiden kirjoittamisessa ja simuloinnissa. Jatkuvissa mittauksissa syntyy tarve automatisoida datan käsittely. Useita satoja tai tuhansia tiedostoja analysoivan ohjelman teko on Matlabissa verraten helppoa ja säästää valtavasti aikaa. Lisäksi hieman nopeammissa mittauksissa syntyy niin paljon dataa, että sen käsittely Excelissä on käytännössä mahdotonta. Esim. 50 Hz mittaustaajuudella vuorokauden mittauksessa kertyy n. 4,2 miljoonaa datapistettä eli n. 63 Excelin täyttä saraketta.

Matlabissa on graafisia työkaluja mm. tiedostojen avaamiseen ja kuvaajien piirtämiseen. Lisäksi ohjelmaan on saatavissa suuri määrä erilaisia "toolboxeja" esim. käyrän sovitusta, tilastoanalyyseja, mittauksia ja neuroverkkomalleja varten. Helsingin yliopistolla on kattava Matlab lisenssi (eli sisältää myös runsaasti toolboxeja), joka kattaa ohjelman käytön laitosten koneissa. Lisäksi Matlabiin kuuluu graafinen mallin rakennusympäristö ”Simulink”, jota ei käsitellä tässä oppaassa mutta verkosta löytyy runsaasti ohjeita myös Simulinkille.

Koodi on tekstissä värillistä ja kommentit on merkitty %−merkillä

[1] MATLAB http://www.mathworks.com

2 Ohjelman hankinta ja käytön aloitus

Kuten edellä on mainittu niin Matlab on kaupallinen ohjelma. Tietoa Matlabin hankinnasta löytyy osoitteesta www.mathworks.com. Helsingin yliopiston lisenssi ei valitettavasti sisällä opiskelijoiden kotikäyttöä, mutta kotona Matlabia voi käyttää verkon välityksellä ottamalla ssh-yhteyden yliopiston Unix palvelimelle ruuvi.helsinki.fi, unix tunnukset saat laitoksen ATK-yhdyshenkilöltä. Asennettuasi ohjelman se löytyy Windowsin käynnistä valikosta: Matlab kansiosta ja unixissa/linuxissa kirjoittamalla komentoriville matlab.

3 Muuttujat ja peruslaskutoimitukset

3.1 Muuttujien määrittely ja peruslaskutoimitukset

Peruslaskutoimitusten teko on helppoa. Muuttujia ei tarvitse erikseen määritellä vaan määrittely tapahtuu automaattisesti, kun muuttujalle annetaan arvo. Eli yksinkertaisesti kirjoitetaan komentoriville(ikkunaan):

```
%Muuttuja a saa arvon 1
a=1
%Muuttuja b saa arvon 2
b=2
```

Tämän jälkeen voidaan suorittaa peruslaskutoimitus määritellyillä muuttujilla. Huomaa, että muuttujien kirjankoolla on väliä eli $a \neq A$.

```
c=a+b
c=a-b
c=a*b
c=a/b
```

Huomaat, että ohjelma tulostaa vastauksesi ruudulle. Yksittäisissä laskutoimituksissa tämä ei haittaa, mutta suurilla datamäärillä tulosta ei haluta aina nähdä. Laskutoimituksen tulos saadaan piilotettua lisäämällä ; komennon perään.

Tulos saadaan jälleen ruudulle kirjoittamalla muuttujan nimi. Kaikki määrittelemäsi muuttujat tallentuvat ohjelmien työtilaan Matlabissa se on näkyvissä ohjelman käyttöliittymässä. Saat listattua muuttujat työtilaan komennolla who ja työtilan saa tyhjennettyä komennolla clear.

```
%Näyttää kaikki työtilassa olevat tiedostot
```

```
who
%Tyhjentää kaikki muistissa olevat muuttujat
clear
%Tyhjentää ruudun
clc
```

4 Vektorit ja Matriisit

Ohjelmat käsittelevät dataa matriisimuodossa tai vektorimuodossa. Muuttuja, jossa on vain yksi rivi tai sarake on vektori ja muuttuja, jossa on monta saraketta ja riviä on matriisi (array). Myös matriisimuotoisilla muuttujilla voidaan suorittaa skalaarilaskutoimituksia ja tässä oppaassa keskitytäänkin lähinnä niihin. Seuraavassa muutamia esimerkkejä laskemisesta vektoreilla ja matriiseilla.

4.1 Vektorilaskutoimituksia

```
%Luodaan vektori, joka saa arvot 1-5}
vektori=1:5
vektori=[1 2 3 4 5]
%Yhteen ja vähennyslasku
vektori+vektori
vektori-vektori
vektori+5
vektori-5
%Kerrotaan koko vektori 2:lla
tupla=vektori*2
%Skalaaritulo. HUOMAA piste "vektori":n jälkeen kertoo
%ohjelmalle että kyseessä on skalaarilaskutoimitus
vektori.*tupla
%Vektoritulo
vektori*tupla
%Jakolasku
vektori./tupla
%Korotetaan toiseen
vektori.^2
luku="vektorin" neljäs numero
luku=vektori(4)
%Vektori "luvut" = "vektorin" 3:s-5:s numero
luvut=vektori(3:5)
```

4.2 Matriisien käsittelyä

Matlabin vahvuus datan käsittelyssä on datan käsittely matriisimuodossa, joka tekee suurenkin datamäärän käsittelyn joustavaksi. Yleensä perusmittausdatan käsittelyssä ei tarvita osaamista matriisilaskennassa, vaan halutaan esimerkiksi, kertoa ja jakaa koko datamäärä esim. kalibrointikertoimellatms. tämä käy komentoriviltä hyvin helposti. Usein laskutoimituksia halutaan suorittaa myös riveittäin eli esim. laskea jokaisen rivin summa, keskiarvo, hajonta... Tämä esitellään seuraavassa kappaleessa.

```
%Luodaan 3x2 matriisi
matriisi=[3 2; 4 1; 6 3]
%Kerrotaan kaikki matriisin arvot 2:hdella
tupla=matriisi*2;
%Skalaaritulo
tupla.*matriisi
% Transponoidaan matriisi
transmatriisi=matriisi'
%Matriisitulo
matriisi*transmatriisi
matriisi*matriisi'
```

Jne. Käyttäytyvät samoin kuin vektorit.

```
%Valitaan matriisin ensimmäinen rivi (1) ja kaikki
%sarakkeet
rivi=matriisi(1,:)
%Valitaan matriisin kolmas(3) sarake ja kaikki rivit(:)
sarake=matriisi(:,3)
%Valitaan matriisin rivit 1–2 (1:2) ja sarakkeet 2–3
%(2:3)
osamatriisi=matriisi(1:2,2:3)
%Matriisin koko
size(matriisi)
```

5 Datan peruskäsittely ja kuvaajat

Tässä kappaleessa tutustutaan data peruskäsittelyyn. Aloitetaan luomalla data satunnaisluvuista ja laskemalla sen keskiarvo mean ja keskihajonta std. Ohjelma palauttaa ruudulle datan jokaisen sarakkeen keskiarvon ja keskihajonnan.

```
%Luodaan data (10x5 matriisi) satunnaisluvuista välillä
%0-1 ja kerrotaan se 10:llä
data=rand(10,5);
%Keskiarvo
mean(data)
%Keskihajonta
std(data)
%Tallentaa keskiarvon muuttujaan ``keskiarvo``
keskiarvo=mean(data);
```

Usein datasta halutaan myös kuvaaja. Ohjelmassa on mahdollisuus piirtää useita erilaisia kuvaajia, alla muutamia yksinkertaisia esimerkkejä.

```
%Viivakuvaaja
plot(x,y)
%Viivakuvaaja, viivan leveys, tyyppi katkoviiva ja väri
%punainen ks. seuraava esimerkki
plot(x,y,'--r','linewidth',2);
%Pylväsdiagrammi
bar(data)
%Piirretään histogrammi eli jakauma
hist(data)
%Piirakkadiagrammi
pie(data)
```

Kuvien muokkaaminen onnistuu Matlabissa suoraan kuvaajan ikkunasta "plot tools":n avulla, mutta selitteiden lisääminen on usein kätevämpää komentoriviltä. Muutamia peruskomentoja:

```
xlabel('x-akseli')
ylabel('y-akseli')
title('kuvaajan otsikko')
axis([x-pienin x-suurin y-pienin y-suurin])
```

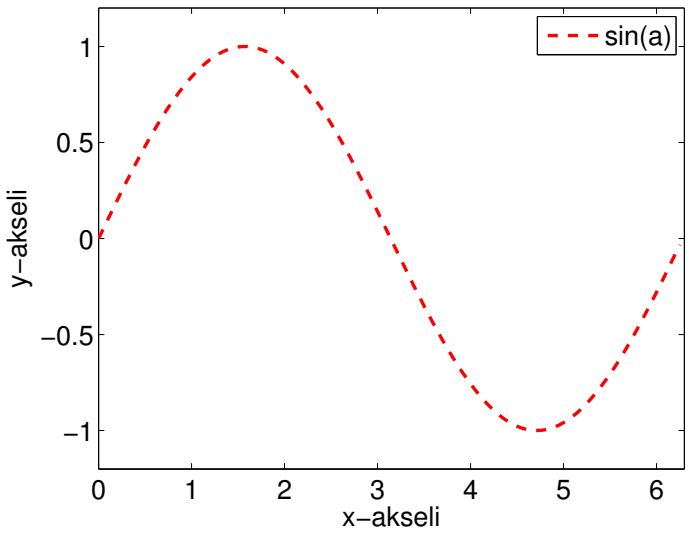
Kuvan tallentaminen tiedostoon onnistuu käskyllä print, tiedostomuoto määritellään syntaksilla ”-dmuoto”. Kuvan voi tallentaa myös valikosta ”file->save as” tai ”file->export setup”. Esimerkkejä komentorivi:

```
%png
print -dpng kuva.png
%värillinen postscript
print -dpsc  kuva.ps
%pdf
```

```
print -dpdf kuva.pdf
%encapsulated postscript
print -dpepsc kuva.eps
```

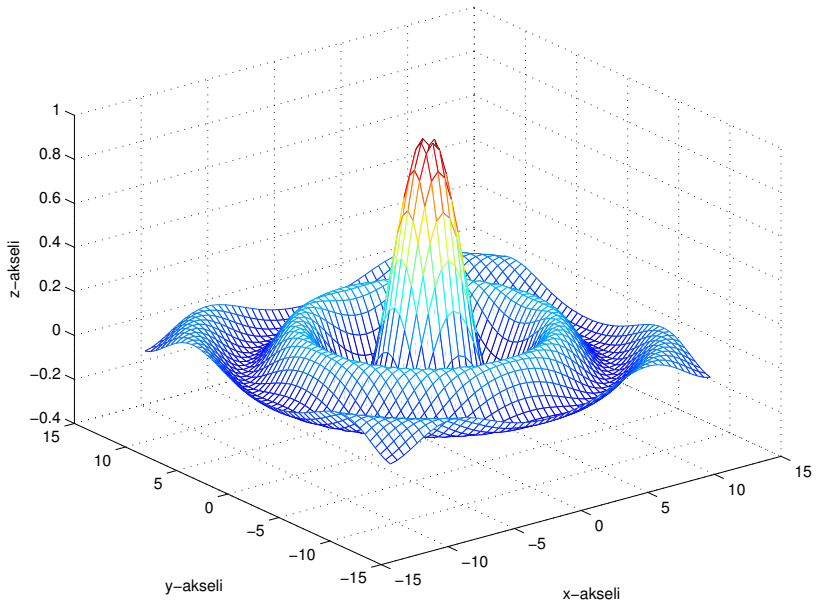
5.1 Esimerkkiohjelma 2d kuvaaja ja sen tallennus.

```
%Sinikäyrän piirto ja tallennus png tiedostoon
a=0:0.05:2*pi;
plot(sin(a),'--r','linewidth',2);
legend('sin(a)')
%Merkinnät ja akselien rajat
xlabel('x-akseli'); ylabel('y-akseli')
axis([0 6.3 -1.2 1.2])
%Tallennus png-formaatissa
print -dpng sini.png
```



5.2 Esimerkkiohjelma 3d kuvaaja ja sen tallennus

```
%Luodaan matriisit x ja y meshgrid funktiolla
[x,y]=meshgrid([-12:1/2:12]);
%Lasketaan z-koordinaatti eri funktion arvoilla
z=sin(sqrt(x.*x+y.*y))./sqrt(x.*x+y.*y);
%Piirretään kuvaaja
mesh(x,y,z);
%Merkinnät kuvaajaan
title('Mesh')
xlabel('x-akseli')
ylabel('y-akseli')
zlabel('z-akseli')
%Tallennetaan kuva3d.eps tiedostoon
print -depsc kuva3d.eps
```



5.3 Julkaisukelpoiset kuvat

Julkaisukelpoisten kuvien tuottaminen Matlabilla on helppoa. Kuvia pystyy muokkaamaan ”Plot toolsien avulla”. Työkalut saa näkyviin klikkaamalla kuvan päällä näkyvistä kuvakkeista oikean puolimmaisinta ”Show plot tools”. Voit helposti muokata kuvaajan värejä, fonttia, viivan paksuutta, lisätä selitteitä ja myös piirtää kuvaan mm. nuolia. Ohjelmasta löytyy myös erittäin kätevä toiminto, jolla kuvan saa tallennettua haluamaansa formaattiin oikeassa koossa ja sopivalla resoluutiolla. Se löytyy valikosta ”File->Export Setup...”. Avautuvasta ikkunasta voit muut-

taa haluamiasi ominaisuuksia ja lopuksi tallennettua kuvan valitsemalla ”Export”. Julkaistavia kuvia muokatessa, ehkä tärkeintä on valita riittävä resoluutio eli ainakin 300, mutta mieluummin 600 dpi (löytyy ”rendering” - välilehdeltä). Hyviä laadullisesti häviöttömiä formaatteja kuvien tallennukseen ovat eps, pdf, tiff ja png (toisin kuin esim. jpg ja gif).

6 Tiedostojen käsittely ja hakemistot

Matlab tunnistaa dos- ja unix-komentoja hakemistoissa liikkumiseen ja tiedostojen käsittelyyn. Hakemistoissa liikkuminen tapahtuu komennolla cd hakemisto.

```
cd d:\data
cd c:\
```

Hakemistossa olevat tiedostot saadaan listattua ruudulle komennoilla

```
ls
dir
% Listaa kaikki tekstitiedostot hakemistossa
ls *.txt
dir *.txt
tiedostot=dir('*.txt');
```

Numeroarvoja sisältävä tiedosto saadaan ladattua ohjelmaan komennolla load tai komennolla dlmread Matlabin valikoista löytyy myös työkalu nimeltä Import wizard. Excel tiedostoja voi lukea komennolla xlsread.

Esim. luetaan tiedoston "data.txt"sisältö muuttujaan "data". Puolipiste komennon viimeisenä kertoo ohjelmalle, että kaikkia numeroita ei tulosteta näytölle. Huomioi, että desimaalierottimena toimii ainoastaan piste, jos siis olet käyttänyt pilkkua ohjelma ei osaa lukea numeroita ja on syytä käyttää esim. notepadin korvaa toimintoa pilkkujen muuntamisessa pisteiksi. Ohjelmien omassa formaatissa olevien tiedostojen avaaminen on vielä helpompaa:

```
%Toimii jos tiedostossa on pelkkiä numeroita
data=load('data.txt');
%Luetaan data ohjelmaan jättäen ensimmäinen rivi pois
%(1,0) ja käytetään erottimena sarkainta)
data=dlmread('data.txt','\t',1,0);
%Avataan ohjelman oma tiedosto tiedosto.mat ks.
%tallentaminen alla
```

```
load tiedosto.mat
```

Tiedostojen tallentaminen Matlabin omassa binäärimuotoisessa *.mat formaatissa onnistuu komennolla save. Data on myös mahdollista tallentaa ASCII-muodossa komennolla dlmwrite ja excel muodossa komennolla xlswrite . Tässä muutamia esimerkkejä save käskyn käytöstä:

```
%Tallentaa kaikki työtilan muuttujat tiedostoon
%tiedosto.mat
save  tiedosto.mat
%Tallentaa työtilassa olevat muuttujat "muuttuja1" ja "
%muuttuja2" tiedostoon tiedosto.mat
muuttuja1=1; muuttuja2=2;
save tiedosto.mat muuttuja1 muuttuja2
%Tallennetaan Matlabin 6-version kanssa yhteensopiva
%tiedosto. Tarvitaan, jos halutaan käyttää matlabin
%vanhaa versiota tai esim. Octave ja R ohjelmia
%tiedoston avaamiseen
save -v6  tiedosto.mat
```

7 Omat ohjelmat

Käyttäjän omia ohjelmia Matlabissa ovat m-filet eli *.m päätteiset tiedostot. Niiden avulla pystyään automatisoimaan tiedostojen käsittely tai tekemään esimerkiksi simulointiohjelma. M-filejä on kahta tyyppiä: itsenäisesti komentoriviltä ajettavia ohjelmia eli skriptejä, joilla voi automatisoida tehtäviä ja funktioita, joita voi kutsua komentoriviltä (tai toisista ohjelmista) ohjelman omien funktioiden tapaan, kuten esim. mean - funktiota. Matlabissa on oma editori ja uusi M-file saadaan tehty valitsemalla "File>new>M-file".

7.1 Skriptit

Skripti on sarja komentoja, jotka suoritetaan peräkkäin. Ohjelma voi sisältää myös ehtolauseita, silmukoita ja kutsuja toisiin ohjelmiin. Alla yksinkertainen esimerkki. Kaikki skriptissä määritellyt muuttujat jäävät muistiin ohjelman työtilaan.

```
%Tämän esimerkin tarkoituksena on näyttää miten M-file
%toimii. Ensin määritellään muuttujat ja sitten
%suoritetaan laskutoimitus ja tallennetaan tulos.
%Kirjoita tai kopioi tämä teksti suoraan editoriin ja
```

```
%tallenna esim. nimellä koe.m, johonkin hakemistoon
%missä tiedät sen olevan.

%Tyhjennetään työtila, etteivät aikaisemmat muuttujat
%sotke ohjelmaa
clear;
%x=100x1 vektori, jossa on satunnaislukuja
x=rand(100,1);
%lasketaan x:n keskiarvo
keskiarvo=mean(x)
%lasketaan x:n keskihajonta
keskihajonta=std(x)
%Piirretään x:stä kuvaaja
plot(x)
xlabel('x-akseli')
ylabel('y-akseli')
%Tallennetaan tulos
save tulos.mat
%Tämän jälkeen aja tiedosto siirtymällä kyseiseen
%hakemistoon ja kirjoita m-filen nimi eli "koe.m" ->
%ENTER.
```

7.2 Funktiot

Funktio luodaan samalla tavalla, kuin itsenäisesti ajettava ohjelma, mutta sen alkuun kirjoitetaan määrittely, että kyseessä on funktio. Funktiota kutsutaan myös komentoriviltä kuten skriptiä, mutta funktioon annetaan haluttu määrä argumentteja (dataa, kertoimia tms.) ja myös kerrotaan mitä muuttujia funktion halutaan palauttavan. Erotuksena skriptiin funktio toimii omassa työtilassaan eli funktiossa määritellyt muuttujat eivät jää ohjelman muistiin. Funktiot ovat käteviä usein suoritettavien laskujen automatisoinnissa. Myös pitkän skriptin pystyy lyhentämään tekemällä sen osista funktioita (aliohjelmia), joita kutsuu pääohjelmasta.

Pystyt kutsumaan tekemääsi funktiota mistä tahansa hakemistosta, jos sijoitat sen ohjelman polussa olevaan hakemistoon (tai lisäät oman hakemistosi polkulistaan (search path)). Soveltuvat hakemistot saat listattua komennolla path ja oman hakemiston saat lisättyä polkuun komennella addpath tai valikosta ”File->Set Path”. Ohjelmat eivät tallennna addpath-komennolla lisäämääsi hakemistoa polkuun vaan se pysyy siellä ainoastaan kunnes ohjelma sammutetaan, tästä ongelmasta pääsee lisäämällä komennon ohjelman käynnistyessään ajamaan tiedostoon (matlabrc.m). Samaiseen käynnistystiedostoon voi lisätä muitakin haluamiaan komentoja esim. cd omahakemisto; clc

```
%Tämä on Matlabin esimerkkifunktio "kluvut", joka laskee
%siihen
%syötetyn datan keskiarvon, hajonnan ja keskiarvon
%keskivirheen. Tallenna tiedosto aina samalla nimellä,
%mikä on
%funktion nimi eli nyt kluvut.m Kutsu komentoriviltä
%muotoa
%[keskiarvo, keskihajonta,keskivirhe]=kluvut(arvo) HUOMAA
%, että
%"keskiarvo, keskihajonta, keskivirhe ja luvut" ovat
%muuttujan nimiä
%ja niiden tilalla voidaan käyttää haluttuja nimiä.
%Muuttujassa "arvo"
%on käsiteltävä data, "keskiarvo" keskihajonta ja
%keskivirhe" ovat
%funktion tuloksia.

function [keskiarvo,keskihajonta,keskivirhe]=kluvut(arvo)
    keskiarvo=mean(arvo);
    keskihajonta=std(arvo);
    %Lasketaan datassa olevien arvojen määrä, eli n
    n=length(arvo);
    %Keskiarvon keskivirhe on keskihajonta/sqrt(n)
    keskivirhe=keskihajonta/sqrt(n);
```

Kun olet tehnyt funktion siirry siihen hakemistoon mihin se on tallennettu ja koita, että funktio toimii: Luodaan funktiolle annettavat arvot, 1000 satunnaislukua

```
data=rand(1000,1);
%Kutsutaan äsken tehtyä funktiota kluvut komentoriviltä
%ja tallennetaan tulokset muuttujiin x, y ja z
[x,y,z]=kluvut(data)
```

8 Perusohjelmointi

8.1 Silmukka eli loop

Yksi lähes joka ohjelmassa tarvittava komponentti on silmukka eli ”loop”. Silmukan sisällä olevat toiminnot toistetaan jokaisella ajokerralla. Silmukkaa tarvitaan esimerkiksi simuloinnissa tai jos halutaan suorittaa lukea monta tiedostoa ohjelmaan ja analysoida ne samalla tavalla. Silmukka ajetaan niin monta kertaa,

kunnes annettu pysäytysehto täyttyy. Yleensä toiminta saadaan suoritettua käyttämällä for - looppia, joka ajaa komennot tietyn määrän kertoja. Lue lisää matlabin ohjeesta.

```
%Esimerkki ajetaan silmukka 10 kertaa, muuttuja kerta saa
% jokaisella ajokerralla uuden arvon. Ajamalla ohjelman
%saat käsityksen mitä tapahtuu loopissa..

%Aloitetaan silmukka
for kerta=1:10
    kerta
%Rakennetaan loopilla vektori, joka saa aina arvon kerta
%*2
    vektori(kerta)=kerta*2
%Lopetetaan silmukka
end
```

8.2 Ehtolauseet: if, else ja elseif

Silmukoiden lisäksi ohjelmissa tarvitaan usein ehtolauseita, joilla kontrolloidaan tiettyjen käskyjen suorittamista. Ehtolauseet toteutetaan komennoilla if, else ja elseif . if ja elseif komennot päätetään komennolla end. Yksinkertaisia esimerkkejä:

```
%b=3 jos a>4
if a>4
    b=3
end

%b=3 jos a>3 ja b=6 jos a<2, muussa tapauksessa b=0
if a>3
        b=3
            else if a<2
            b=6
            else
                b=0
          end
end
```

9 Tehtävien automatisointi eli ”Batch-mode”

Usein mittauksissa kertyy suuri määrä samanlaisia tai lähes samanlaisia tiedostoja joiden käsittely halutaan tehdä mahdollisimman automaattisesti. Tämä tapahtuu tekemällä oma peräkkäisistä komennoista koostuva ohjelma, jolla voi vaikkapa laskea kaikkien hakemistossa olevien tiedostojen keskiarvon.
Esimerkkinä käytän lehmän jalkapainomittauksesta saatua dataa, jonka voi ladata internetistä http://files.mpastell.com/hdata.zip. Myös ao. koodi löytyy samasta zipistä nimellä ”hdata.m”. Data on saatu mittaamalla lehmän jokaisen jalan painoa erikseen lypsyn aikana ja tiedosto koostuu neljästä sarakkeesta, joista jokaisessa on yhden jalan paino. Esimerkissä lasketaan jokaisen jalan keskipaino ja keskihajonta lehmän kokonaispaino, ja tallennetaan jokaisesta tiedostosta kuvaaja erilliseen hakemistoon.

```
%Datassa on otsikkorivi ja itse mittaustieto 7:ssä
%sarakkeessa. Harjoituksessa käytetään sarakkeita 2-5,
%joissa on lehmän jokaisen jalan paino lypsyn aikana.
clear; clc;
%Luetaan kaikki hakemiston tekstitiedostot muuttujaan ``
%tiedostot``
tiedostot=dir('*.txt');
%Lasketaan tiedostojen määrä ja luodaan hakemisto kuville
maara=length(tiedostot);
mkdir('kuvat')
%Ajetaan silmukka, jossa käsitellään kaikki hakemistossa
%olevat tiedostot
for kerta=1:maara
    %Luetaan silmukan ajokerran osoittama tiedosto
    data=dlmread(tiedostot(kerta).name,'\t',1,0);
    %Valitaan kiinnostuksen kohteena oleva sarakkeet 2-5
    data=data(:,2:5);
    %Lasketaan lehmän kokonaispaino jalkapainojen summana
    data(:,5)=sum(data,2);
    %Lasketaan tiedoston jokaisen sarakkeen keskiarvo.
    %Lopuksi muuttujassa on kaikkien tiedostojen
    %keskiarvot, jokainen omalla rivillään.
    keskiarvo(kerta,1:5)=mean(data);
    %Lasketaan keskihajonta
    keskihajonta(kerta,1:5)=std(data);
    %Piirretään kuvaaja ja lisätään selitteet
    plot(data)
    xlabel('measurement point')
```

```
    ylabel('weight (kg)')
    %Tulostetaan kuva png-tiedostoon.
print(['kuvat/' num2str(kerta) '.png'],'-dpng')
end
```

10 Sekalaisia vinkkejä

Tämän otsikon alla on muutamia esimerkkiohjelmia ja mielestäni hyödyllisiä komentoja.Tämän osion esimerkkejä on kommentoitu vähemmän kuin alkuosaa, tarkoituksena on myös että niitä soveltaessa pitää ymmärtää mitä tekee.

10.1 Hyödyllisiä komentoja

1. find komennolla voidaan etsiä vektorista tai matriisista kaikki tietyn ehdon täyttävät arvot. Mielestäni hyödyllisin komento, säästää valtavasti laskenta-aikaa jos käytetään loopin sijaan (Ks. seuraava kappale). Esim. haetaan muuttujasta ”data”, kaikki yli 3*keskihajonta keskiarvosta poikkeavat arvot ja poistetaan ne.

```
data=[2 3 4 3 3 2 4 4 2 3 3 3 3 4 4 2 2 50];
sd=std(data);
ka=mean(data);
pois=find(data>ka+3*sd | data<ka-3*sd);
data(pois)=[];
```

2. diff(data) käsky laskee vektorin ”data” peräkkäisten elementtien välisen erotuksen. Esimerkki seuraavassa kappaleessa.

3. tic ja toc komennot mittaavat komentojen välissä kulunutta aikaa. Lisäämällä komennot ohjelmaan alkuun ja loppuun saadaan selville ohjelman suorittamisessa kulunut aika.

4. pause(1) lisää ohjelmaan 1 sekunnin mittaisen tauon.

5. sortrows(data,[1 2]) järjestää matriisin rivit ensiin sarakkeen 1 mukaan ja sitten sarakkeen 2 mukaan.

6. vektori(10)=[] poistaa vektorin 10:n elementin

7. [a b] yhdistää vektorit tai matriisit a ja b.

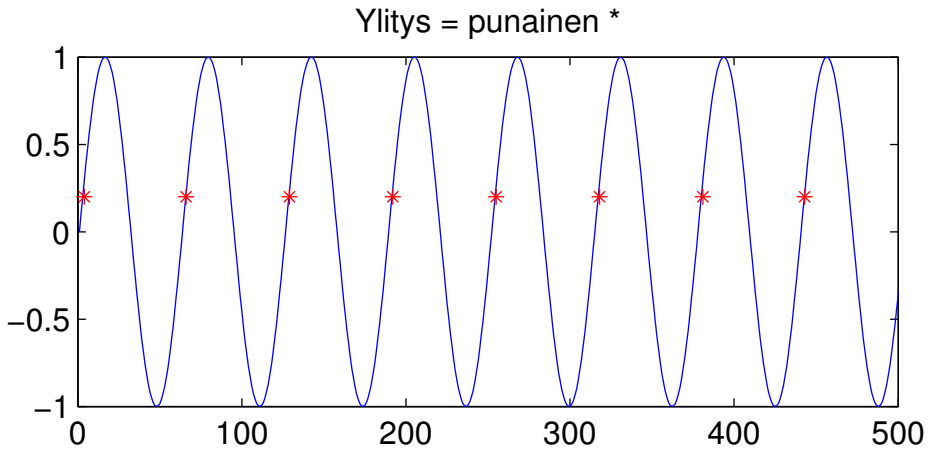
10.2 Piikkien etsintä datasta

Seuraavilla esimerkeillä lasketaan kuinka monta kertaa data ylittää tietyn raja-arvon. Samalla tavalla laskea minkä tahansa piikkien määrää datassa. Laitoksella samantyyppistä laskentaa on hyödynnetty mm. lehmän potkujen määrän laskemiseksi lypsyn aikana jalkapainodatasta[1,2] ja lehmän hengitystaajuuden laskennassa lypsyn aikana.[3]

```
%Lasketaan kuinka monta kertaa kuvaaja ylittää arvon 0.2
clear; clc;
data=sin(0:0.1:50);
%Raja-arvon ylittavät pisteet
pyli=find(data>0.2);
%Kerrat jolloin raja-arvo ylitetään
yli=find(diff(pyli)>1);
kerrat=length(yli);
%Ensimmäinen kerta jää laskematta jos alkuarvo<0.2;
if data(1)<0.2 kerrat=kerrat+1; end
plot(data)
title(['Ohjelma laski ' num2str(kerrat) 'ylitystä.'])
```

Rajan ylitykset voi laskea myös hyödyntämällä pelkästään find käskyä. Ohjelma on myös helpohkosti muokattavissa huippujen haku ohjelmaksi:

```
%Lasketaan kuinka monta kertaa kuvaaja ylittää arvon 0.2
clear; clc;
data=sin(0:0.1:50);
data2(2:length(data)+1)=data;
data(length(data)+1)=NaN;
yli=find(data>0.2 & data2≤0.2);
kerrat=length(yli);
y(1:kerrat)=0.2;
plot(data);
hold on
plot(yli,y,'*r');
title('Ylitys = punainen *')
```



10.3 Tekstitiedostojen muunto *.mat formaattiin

Joskus on hyödyllistä muuntaa ASCII - muotoista (txt) dataa Matlabin omaan dataformaattiin. Tähän on 2 syytä: tiedostot vievät paljon vähemmän tilaa ja aukeavat ohjelmissa suunnattoman paljon nopeammin. Alla oleva ohjelma muuntaa kaikki hakemistossa olevat *.txt tiedostot *.mat tiedostoiksi säilyttäen tiedostonimen muuten ennallaan. Uuteen *.mat tiedostoon tallennnetaan tiedostossa oleva data muuttujaan ”data” ja tiedoston tallennuspäivämäärä muuttujaan ”date”.

```
clc; clear;
%Listataan hakemiston *.txt tiedostot struktuuriin files
files=dir('*.txt');
tlkm=size(files,1)

for i=1:tlkm
    %Käytetään dlmread komentoa, jos datassa on
    %otsikkorivi,load komentoa jos ei ole (käyttäjä
    %valitsee).
    %data=dlmread(files(i).name,'\t',1,0);
    data=load(files(i).name);
    date=files(i).date;
    nimi=strrep(files(i).name, 'txt', 'mat');
    save(nimi, 'data','date')
end
```

10.4 Matlabin dialogit

Matlabin ohjelmiin voi lisätä myös mm. graafisia dialogeja, joilla voidaan esimerkiksi seurata silmukan edistymistä (tämä on joskus mukava ominaisuus hieman enemmän aikaa vievässä ohjelmassa) tai valita avattava/tallennettava tiedosto. Lisätietoa löytyy Matlabin helpistä kohdasta ”Predefined Dialog Boxes”. Esimerkkiohjelma luo silmukan edistymistä seuraavaan näytön.

```
%Esimerkki 'waitbar' käskystä.
h=waitbar(0,'Odota...');
for i=1:100
        pause(0.1)
        waitbar(1/100)
end
close(h)
```

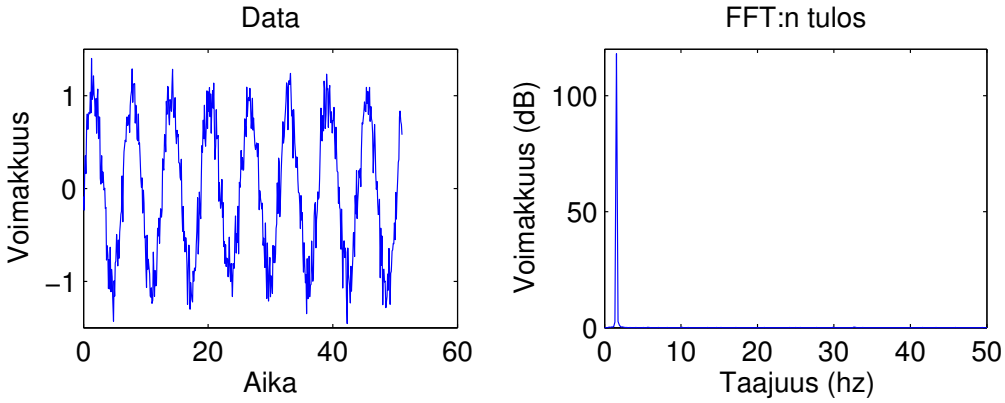
11 Signaalin käsittely

11.1 FFT - taajuuden etsintä signaalista

FFT (Fast Fourier Transform) on laskentamenetelmä, jolla voidaan laskea signaalissa esiintyvien taajuuksien voimakkuus. Alla esimerkki FFT:n laskemisesta Matlabilla.

```
clear; clc;
%FFT:llä voidaan laskea eri taajuuksien voimakkuus
%signaalissa. Tämä on esimerkki FFT:n käytöstä
%Matlabissa. Datapisteet 512 kpl, mittaustaajuus 100hz
data=(0:0.1:51.1);
%Luodaan häiriötä dataan
noise=randn(1,512);
%Lisätään häiriö sin(dataan)
hdata=sin(data)+noise/5;
plot(data,hdata)
%Merkinnät ja akselien rajat
title('Data')
xlabel('Aika')
ylabel('Voimakkuus')
%Lasketaan DFT häiriöllisestä datasta fft funktiolla
Y = fft(hdata,512);
```

```
%Lasketaan power spectral density eli eri taajuuksien
%voimakkuus signaalissa
Pyy = Y.* conj(Y) / 512;
%Ainoastaan ensimmäisillä 257 pisteellä on merkitystä.
%Lasketaan taajuusvektori eli taajuus/fft:n pisteiden
%määrä
f = 100*(0:256)/512;
%Plotataan fft:n tulos uuteen kuvaan
figure
plot(f,Pyy(1:257))
xlabel('Taajuus (hz)')
ylabel('Voimakkuus (dB)')
title('FFT:n tulos')
```



12 Neuroverkot

Tämän ja seuraavan kappaleen tarkoituksena on osoittaa, että hienojen termien ”Neuroverkko” ja ”Sumea logiikka” toteutus onnistuu ilman mitään toolboxeja ja melko yksinkertaisilla laskutoimituksilla. Mallien toimintaan perehtyminen kirjallisuuden perusteella voi olla aluksi vaikeaa, koska niissä käytetään paljon omaa terminologiaa ja sinällänsä helpoille laskutoimituksille on annettu hienot nimet. Toivottavasti nämä yksinkertaiset esimerkit hieman selventävät asiaa.

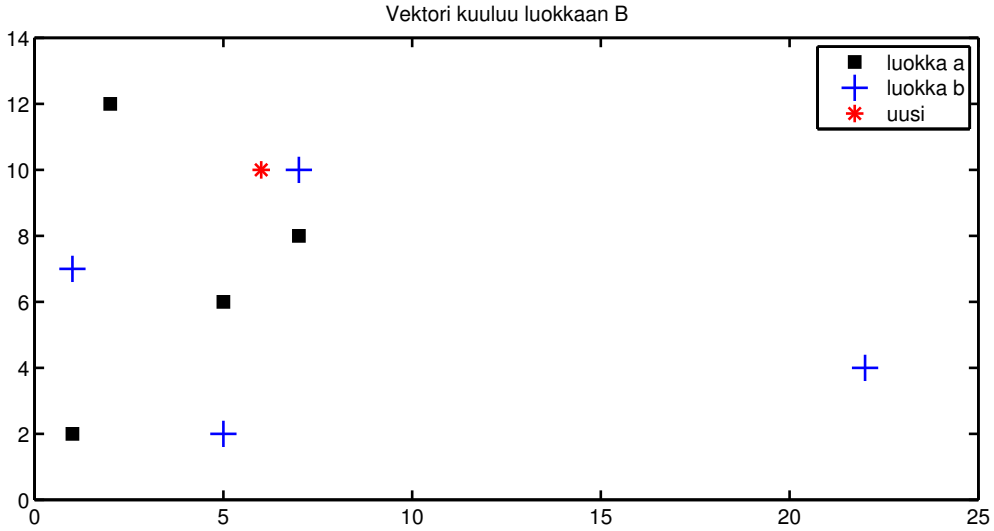
12.1 Probabilistic Neural Network

Probabilistic Neural Network (PNN) on neuroverkkomalli, joka luokittelee uusia vektoreita opetusdatan perusteella. Malli säilyttää koko opetusdatan toisin kuin monet muut samaan tarkoitukseen kehitetyt NV-mallit. Esimerkki mallin toiminnasta:

```
%PNN kahdella luokituksella tässä esimerkissä
%opetusdatana ovat
%keksityt a ja b, mutta ne voidaan helposti korvata
%oikealla
%mittausdatalla. Opetusdatan riveillä on esimerkissä vain
% 2 arvoa,
%koska se on helpointa esittää graafisesti. Arvoja voi
%kuitenkin
%lisätä lähes rajatta. Mallin toimintaa säädetään
%muuttamalla
%spreadin=s arvoa

clear; clc;
%Opetusdata a ja b=PNN:n 2 luokkaa
a=[1 2; 2 12; 5 6; 7 8]
b=[1 7; 22 4; 5 2; 7 10]
%s on smoothing parameter (spread)
s=0.2;
%c on luokilteltava vektori
c=[6 10]
%Lasketaan etäisyydet molempien luokkien jokaiseen
%vektoriin(=riviin)
for i=1:length(a);
    eta(i)=sqrt(sum((a(i,:)-c).^2))
    etb(i)=sqrt(sum((b(i,:)-c).^2))
end
%Radial basis activation function saa arvon 1, kun
%etäisyys on 0 ja muulloin<1. Aktivointi funktion "
%leveys" riippuu s:n arvosta.
rba=exp(-s*eta.*eta)
rbb=exp(-s*etb.*etb)
%Lasketaan kaikkien radial basis funktioiden summa
%molemmille luokille
fax=sum(rba)
fbx=sum(rbb)
%Plotataan vektorit valitaan isompi luokka ja näytetään
%se kuvassa
plot(a(:,1),a(:,2),'squareblack')
hold on
plot(b(:,1),b(:,2),'+b')
plot(c(:,1),c(:,2),'*r')
legend('luokka a','luokka b','uusi')
```

```
%Luokituksen valinta
if fax>fbx
    title('Vektori kuuluu luokkaan A')
else
     title('Vektori kuuluu luokkaan B')
end
hold off
```



13 Sumea Logiikka

Tämä on tarkoitettu esimerkiksi laskennasta sumean logiikan idean tunteville. Ensin tehdään gaussinen jäsenyysfunktio gausf.m ja sitten lasketaan yksinkertainen esimerkki sumeasta mallista. Mallissa on kolme jäsenyysfunktiota ja kaikilla jäsenyysfunktioilla on mallissa yhtä suuri paino.

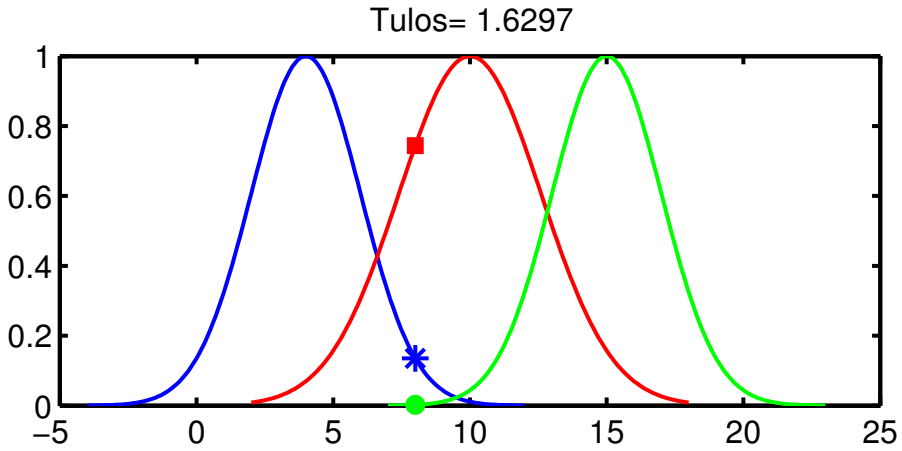
Gaussinen jäsenyysfunktio: $f(x) = e^{\frac{-(x-c)}{2\sigma^2}}$

```
%Jäsenyysfunktio gausf.m, c=jakauman keskipiste, s=
%hajonta eli sigma,x on uusi piste
function mf = gausf(x,c,s)
    mf=exp(-(x-c).^2/(2*s^2));
```

Seuraavaksi malli:

```
%Luodaan sumea malli, jossa on kolme gaussista
%jäsenyysfunktiota. c=funktion keskipiste ja s=hajonta
```

```
%ja a=tulos, kun kuuluu funktioon jäsenyysasteella 1.
%Lopuksi lasketaan pisteen p arvo mallilla
clc; clear;
%Määritellään jäsenyysfunktiot
c1=4; s1=2; a1=1;
c2=10; s2=2.6; a2=2;
c3=15; s3=2; a3=3;
%Piirretään kuvaajat
pisteet=-8:0.1:8;
mf1=gausf(c1+pisteet,c1,s1);
mf2=gausf(c2+pisteet,c2,s2);
mf3=gausf(c3+pisteet,c3,s3);
plot(pisteet+c1,mf1,'b');
hold on
plot(pisteet+c2,mf2,'r');
plot(pisteet+c3,mf3,'g');
%Lasketaan pisteen p arvo ja piirretään jäsenyysasteet
%kuvaan
p=8;
j1=gausf(p,c1,s1);
j2=gausf(p,c2,s2);
j3=gausf(p,c3,s3);
plot(p,j1,'*b')
plot(p,j2,'xr')
plot(p,j3,'og')
tulos=j1*a1+j2*a2+j3*a3;
hold off
title(['Tulos= ' num2str(tulos)])
```



Viitteet

[1] Pastell, M., H. Takko, H. Grohn, M. Hautala, V. Poikalainen, J. Praks, I. Veermae, M. Kujala and J. Ahokas. 2006. Assessing cows' welfare: Weighing the cow in a milking robot. Biosystems Engineering. 93:81-87.

[2] Pastell, M., H. Takko, H. Grohn, M. Hautala, V. Poikalainen, J. Praks, I. Veermae, M. Kujala and J. Ahokas. 2006. Assessing cows' welfare: Weighing the cow in a milking robot. Biosystems Engineering. 93:81-87.

[3] Pastell, M., A. M. Aisla, M. Hautala, J. Ahokas, V. Poikalainen, J. Praks and I. Veermae. 2006. Automatic cow health measurement system in a milking robot. in ASABE annual international meeting, Portland, USA.

Hakemisto