# Matlab tutorial, 02.05.06 & 03.05.06

## Vectors and matrices

```
Y=[1 2 3]
Y = [1 2 3];
Z=Y'                              % transpose of matrix Y
Z*Y
Y*Z
Y.*Y
Y.^2
Y^2
W=Z*Y
W^2
W.^2
W(1,1)=2; W(2,2)=4; W(3,3)=8;  % change elements of matrix W manually
W
det(W)
inv(W)
B=2*ones(3,1)                     % ones(n,m) creates an (nxm) - identity matrix
X=W\B                             % solves the linear equation WX=B
inv(W)*B
W=W(1:2,:)                        % choose the first and second row of matrix W
B=B(1:2)
Y=W\B                             % if W is not a square matrix, the command solves a least-squares
                                  problem, i.e. it minimizes || WY - B ||^2 = min - compare MATLAB help
                                  for details.
```

## Loading and saving data

load BASFR
save TEMP BASFR

*Here, the data BASFR are saved in a file* TEMP.MAT*; you can save more than one variable in the same file by using the command* save TEMP BASFR BMWR DAXR …
*The command save* TEMP *saves all variables in the workspace in the file* TEMP.MAT.

## Plotting data

plot(BASFR)
plot(BASFR,'*r')

*You can manipulate a figure by using the* Edit *commands of the window in which the figure is displayed. Just play around with the possibilities. Using the* File *commands, you can save the figure - e.g. as file* BASFR.FIG *- in MATLAB's own graphic format. You can* open *and manipulate it later. You also can* export *a figure into some other graphics format for insertion into a paper.*

```
t=1:1:746;
t=t';
plot(t,BASFR)
t=t.^2;
plot(t,BASFR)
load DAXR;
plot(t,BASFR,t,DAXR)
plot(t,[BASFR DAXR])   % the column vectors BASFR and DAXR are merged to a matrix with 2
                       columns.
subplot(2,1,1)         % breaks the Figure window into a 2-by-1 matrix of small axes, refer to first
                       axis.
plot(BASFR)
subplot(2,1,2)         % refer to second axis
plot(DAXR)
```

*There are also* SURF *and* MESH *commands for the 3D-plot. Check it in the MATLAB help.*

## Plotting functions

```
x=-3.5:0.05:3.5;
p=exp(-x.^2/2)/sqrt(2*pi);          % Is it familiar to you?
plot(p)
plot(x,p)
f=normpdf(x,0,1);                   % Returns the normal probability density function with mean 0, and
                                    standard  deviation 1, at the values in x
hold on                             % Keep the previous plot .
plot(x,f)
f2=normpdf(x,1,0.5);
plot(x,[f;f2])
```

## Generating random variables

```
z=normrnd(0,1,200,1);               % Generate N(0,1)-distributed random variables
m=mean(z)
s=std(z)
normplot(z)                         % It is linear if z is normally distributed
help normplot
hist(z,15)
hist(z,21)
histfit(z,21)                       % Fit a normal density function to the histogram
```

## Kernel estimate of probability density

*The following commands calculate an estimate* pe *of the probability density* p *of a sample of iid random variables (in vector* z *– in our case standard normal). The theory is developed in the lecture on nonparametric statistics. At the moment, you have to believe that it is consistent if the sample size* n *goes to infinity and the smoothing parameter* h *goes to* <u>*0*</u> *such that* nh *goes to infinity.*

```
n=length(z)
nx=length(x);
h=0.2;
pe=p;
for i=1:nx
u=normpdf((x(i)-z)/h); pe(i)=sum(u)/(n*h); end
plot(x,[p;pe])                % plot of the true density and the estimate – repeat the calculation with
                              different values of h and look what happens.
pe2=kernpde(z,x,0.1);
plot(x,[p;pe;pe2])
```

*As a second example, consider a mixture of two normal distributions with true density* p

```
p=0.8*normpdf(x,0,0.5)+0.2*normpdf(x,2,0.2);   % the corresponding random variables can be written
                                               as B*Z1+(1-B)*Z2 where Z1, Z2 are variables from the
                                               two normal laws and B is a Bernoulli, i.e. 0-1-variable.
plot(x,p)
B=ceil(unifrnd(0,1,200,1)-0.2);          % generate B in the pedestrian way using the basic uniform
                                         random number generator
z=B.*normrnd(0,0.5,200,1)+(1-B).*normrnd(2,0.2,200,1);
pe=kernpde(z,x,0.1);
plot(x,[p;pe])
B=binornd(1,0.8,200,1);             % alternatively generate B using the special random generators of the
                                    statistics toolbox
z=B.*normrnd(0,0.5,200,1)+(1-B).*normrnd(2,0.2,200,1);
pe=kernpde(z,x,0.1);
plot(x,[p;pe])
normplot(z)
```

***Your task***: write a function file **pdkernest.m** which compute the probability density estimator, which takes z, x, and h as input.

## Distribution of binomial parameter estimate

*We want to see how the generated binomial random variables distributed is. We also compare it with the normal distribution with the parameter p, s.*

```
x=binornd(50,0.1,5000,1);
x=x/50;
mean(x)
std(x)
sqrt(0.1*0.9/50)            % compute the variance manually
median(x)
iqr(x)                      % compute the interquartile range
 [min(x) max(x)]
hist(x,51)
hist(x,31)
histfit(x,31)
help boxplot
boxplot(x)
s=sqrt(0.9*0.1/50);
z=normrnd(0.1,s,5000,1);
boxplot(z)
boxplot([x z])
skewness(x)
skewness([x z])
```

*What happens if you change 50 to 500? To 1000? Compare with the normal distribution!*

## Chi-square goodness-of-fit test

*H0 : x is normally distributed     vs.     H1: x has another distribution*
*Test statistics: chi2 =* $\sum(z_j - e_j)^2/e_j$
$z_j$ *= number of observations lie in interval j*
$e_j$ *= expected number of observations lie in interval j*
*Reject H0 if chi2> quantile of chi square distribution*

```
intup=0.021:0.02:0.201;     % define the intervals
intup=[intup 0.3];
m=mean(x); s=std(x);
max(x)
s=zeros(11,1);
for i=1:11
s(i)=sum(ceil(x-intup(i))); end
s
z=zeros(11,1);
for i=2:11
z(i)=s(i-1)-s(i); end
z(1)=5000-s(1);
z
m=mean(x);
s=std(x);
e=normcdf(intup,m,s);       % computes the normal cdf with mean m and  standard deviation s at
                            the values in intup.
e(2:11)=e(2:11)-e(1:10);    % find the probability for an observation to be in interval j
e(1)=e(1)-normcdf(0,m,s);
e=5000*e';
[z e]
plot([z e],'*')
chi2=sum((z-e).^2./e);
chi2
chi2inv(0.95,8)
```

*Is x normally distributed?*