MATLAB

Lecture 7

Probability and Statistics

- **Random Numbers**
- **Many probabilistic processes rely on random numbers**

MATLAB contains the common distributions built in

»rand

draws from the uniform distribution from 0 to 1

Uniformly distributed pseudorandom numbers

Generate values from the uniform distribution on the interval [a, b].

```
r = a + (b-a).*rand(100,1);
```

Probability and Statistics

- **Random Numbers**

»randn

draws from the standard normal distribution (Gaussian)

»random

can give random numbers from many more distributions

see doc random for help

the docs also list other specific functions

>> randi(imax)

randi(imax) returns a random integer drawn from the discrete uniform distribution on 1:imax.

Probability and Statistics

- **Random Numbers**

The same sequence of numbers will not be generated unless the same starting point is used. This starting point is called the “seed”. Each time you start Matlab, the random number generator is initialized to the same seed value. The current seed value can be seen using

s = rand('seed')

By setting a seed value, we ensure that the same results will be produced each time the script is executed. The seed can be set to a value (say, 1234) as follows:

rand('seed',1234)

Probability and Statistics

- **Whenever analysing data, you have to compute statistics**

»scores = 100*rand(1,100);

Number of built-in functions :

mean, median, mode, … etc

To group data into a histogram

»hist(scores,5:10:95);

makes a histogram with bins centered at 5, 15, 25…95

»N=histc(scores,0:10:100);

returns the number of occurrences between the specified bin edges0 to <10, 10 to <20…90 to <100.

you can plot these manually:

»bar(0:10:100,N,'r')

Probability and Statistics

Let us simulate Brownian motion in 1 dimension.

Make a 10,000 element vector of zeros

Write a loop to keep track of the particle's position at each time

Start at 0. To get the new position, pick a random number, and if it's <0.5, go left; if it's >0.5, go right. Store each new position in the kth position in the vector

Plot a 50 bin histogram of the positions.

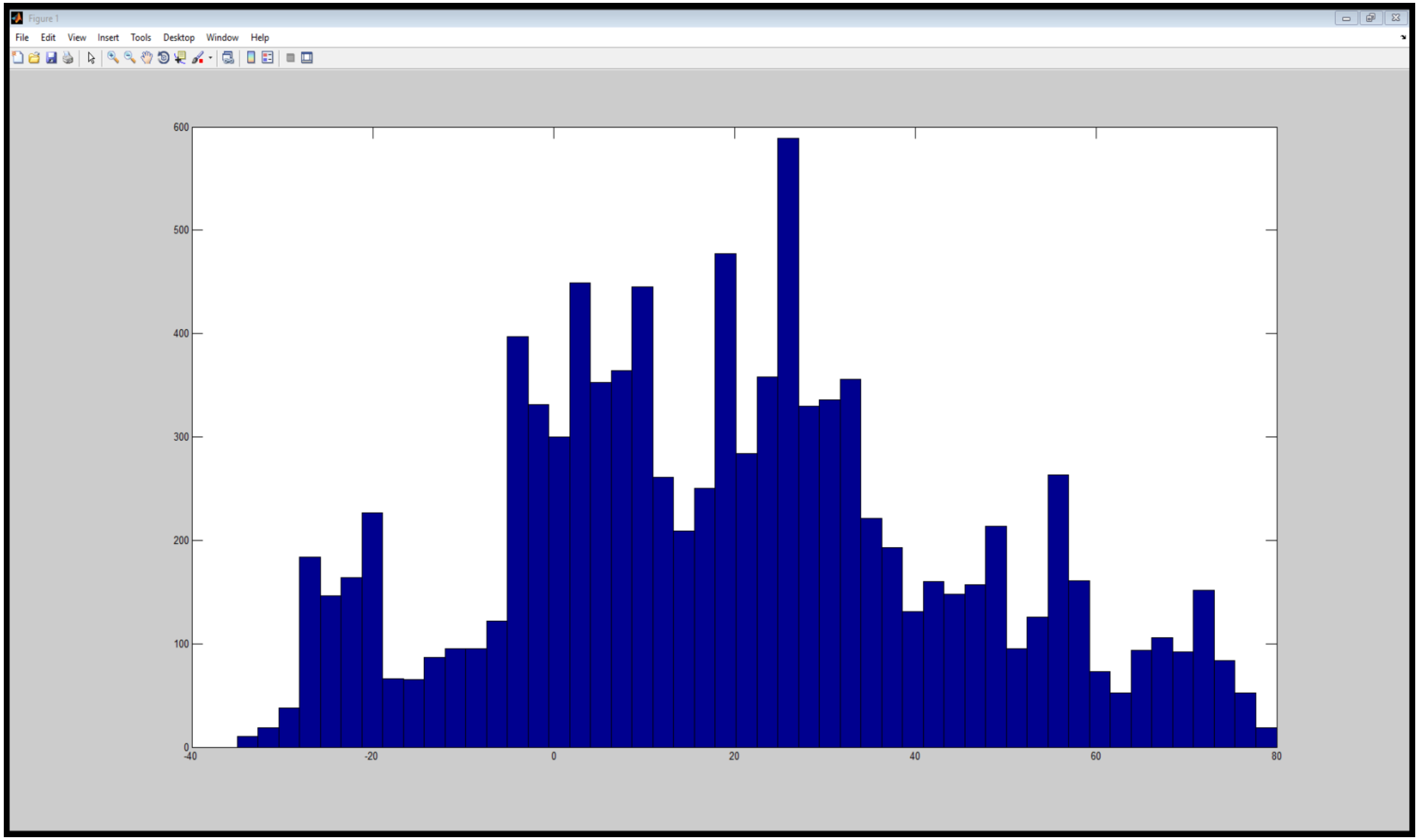
Probability and Statistics

Brownian motion in 1 dimension.

```
x=zeros(10000,1);
for n=2:10000
    if rand<0.5
        x(n)=x(n-1)-1;
    else
        x(n)=x(n-1)+1;
    end
end
figure;
hist(x,50);
```

Probability and Statistics

Brownian motion in 1 dimension.



Probability and Statistics

- **The Normal Distribution**

- The normal distribution-specific **pdf** function contains arguments that has three arguments: a particular x-value at which you seek the **pdf** value, the mean value of the distribution and the standard deviation of the distribution.
- For example, the Matlab command **normpdf(x,xmean,sigma)** returns the value of the probability density function, p(x), evaluated at the value of x of a normal probability distribution having a mean of xmean and a standard deviation of sigma.
- Specifically, typing the Matlab command **normpdf(2,0,1)** yields p(2) = 0:0540.
- A similar result can be obtained using the generic Matlab command **pdf(norm,2,0,1)**

Probability and Statistics

- **The Normal Distribution**

- The normal distribution **cdf** function also has similar arguments: a particular x-value at which you seek the **cdf** value, the mean value of the distribution and the standard deviation of the distribution.
- This is none other than the integral of the pdf function from 1 to x ; P (x).
- So, typing the Matlab command **normcdf(1,0,1)** gives P (1) = 0:8413. This tells us that 84.13% of all normally-distributed values lie between 1 and one standard deviation above the mean.
- Typing **cdf(norm,1,0,1)** yields the same value.

Probability and Statistics

- **The Normal Distribution**

- The normal distribution **"inv"** function gives the inverse of the cdf function, i.e., it provides the value of x for the **cdf** value of P .
- Thus, typing the Matlab command **norminv(0.9772,0,1)** results in the value of 2. This says that 97.72% of all normally-distributed values lie below two standard deviations above the mean.
- There is not an equivalent generic Matlab **inv** command.

Probability and Statistics

- **Student's t Distribution**

- functions tpdf, tcdf and tinv are used for Student's t distribution.
- command tpdf(t,nu) returns the value of the probability density function, p(t; V), evaluated at the value of t of Student's t distribution having a nu degrees of freedom

Probability and Statistics

- probplot(y) creates a normal probability plot comparing the distribution of the data in y to the normal distribution. The plot includes a reference line useful for judging whether the data follows a normal distribution.
- example
- probplot(dist,y) creates a probability plot for the distribution specified by dist, using the sample data in y.

Probability and Statistics

- **Test Data for Normal Distribution Using probplot**
  - Generate sample data containing about 20% outliers in the tails. The left tail of the sample data contains 10 values randomly generated from an exponential distribution with parameter mu = 1. The right tail contains 10 values randomly generated from an exponential distribution with parameter mu = 5. The center of the sample data contains 80 values randomly generated from a standard normal distribution.

Probability and Statistics

- **Test Data for Normal Distribution Using probplot**

```
left_tail = -exprnd(1,10,1);
 right_tail = exprnd(5,10,1);
center = randn(80,1);
data = [left_tail;center;right_tail];.
```

Probability and Statistics

- **Test Data for Normal Distribution Using probplot**

Create a probability plot to assess whether the sample data comes from a normal distribution. Plot a *t* location-scale curve on the same figure to compare with data.

Probability and Statistics

- **Test Data for Normal Distribution Using probplot**

```
figure;
probplot(data);
 p = mle(data,'dist','tlo');
t = @(data,mu,sig,df)cdf('tlocationscale',data,mu,sig,df);
h = probplot(gca,t,p);
h.Color = 'r';
h.LineStyle = '-';
title('{\bf Probability Plot}')
legend('Normal','Data','t','Location','NW')
```

Probability and Statistics

- SIMPLE LINEAR REGRESSION:
  - A bivariate scatterplot is a simple plot of x versus y between two variables.
  - A bivariate scatterplot is a convenient first step to visualize the relationship between the two variables.
  - Assume that we have two variables that are linearly related, except some Gaussian noise term with mean 0 and standard deviation 1
  - $y = 10x + 3 + noise$
  - Assuming that the variable x is a linearly spaced row vector of length 50, between 0 and 1, generate the y vector
  -

Probability and Statistics

SIMPLE LINEAR REGRESSION:

```
randn('seed',1) % specify a seed (optional)
n=50; % number of observations
x=linspace(0,1,n); % linearly spaced vector a length n
 y= 10*x + 3 + randn(1,n);
 plot(x,y,'.')
xlabel('x')
ylabel('y')
```

Probability and Statistics

SIMPLE LINEAR REGRESSION:

In a bivariate scatterplot (x,y), the point with coordinates $(mean(x), mean(y))$, is known as the point of averages.

```
mx=mean(x);
 my=mean(y);
hold on;
plot(mx,my, 'ro', 'markerfacecolor','r')
 legend('data', 'point of averages')
```

Probability and Statistics

SIMPLE LINEAR REGRESSION:

Covariance: Covariance between vectors x and y can be computed in “unbiased” and “biased” versions as

```
c= mean((x-mx).*(y-my)) % covariance (biased)
 n=length(x);
 cs= c*n/(n-1) % sample covariance(unbiased)
```

Probability and Statistics

SIMPLE LINEAR REGRESSION:

Correlation coefficient: The correlation coefficient between two variables is a measure of the linear relationship between them. The correlation coefficient between two vectors can be found using the average of the product of the z-scores of x and y. The “biased” version is

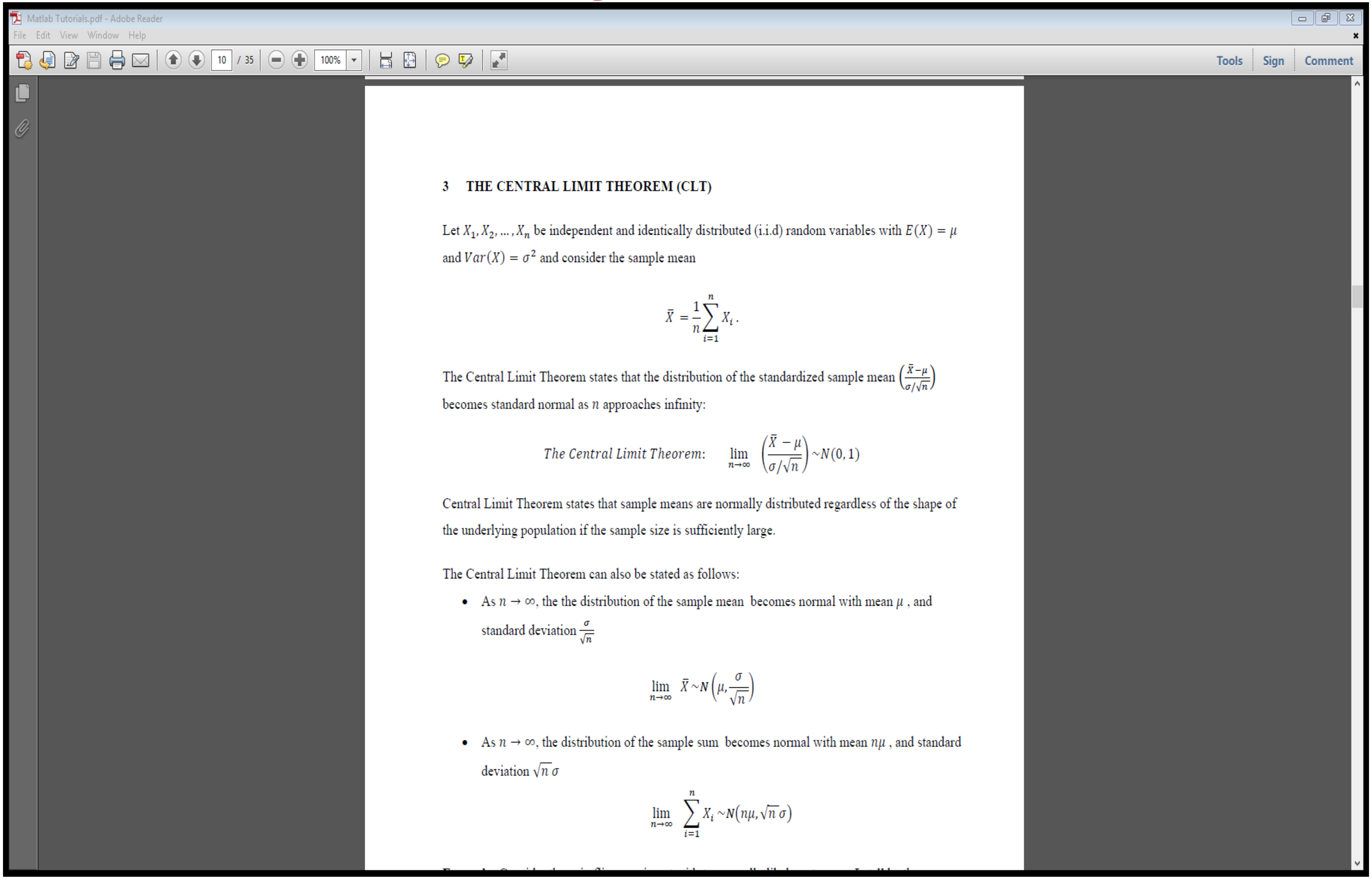
```
zx=zscore(x,1)
 zy=zscore(y,1)
 r=mean(zx.*zy)
```

Correlation coefficient can also be computed from the covariance, as follows:

```
sx=std(x,1);
 sy=std(y,1);
 r=c/(sx*sy)
```

The “unbiased” version (sample correlation coefficient) is computed the same way, except that the flag “1” is replaced by “0”.

Probability and Statistics



3 THE CENTRAL LIMIT THEOREM (CLT)

Let $X_1, X_2, \dots, X_n$ be independent and identically distributed (i.i.d) random variables with $E(X) = \mu$ and $Var(X) = \sigma^2$ and consider the sample mean

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i .$$

The Central Limit Theorem states that the distribution of the standardized sample mean $\left(\frac{\bar{X}-\mu}{\sigma/\sqrt{n}}\right)$ becomes standard normal as n approaches infinity:

$$\textit{The Central Limit Theorem:} \quad \lim_{n\to\infty} \left(\frac{\bar{X}-\mu}{\sigma/\sqrt{n}}\right) \sim N(0,1)$$

Central Limit Theorem states that sample means are normally distributed regardless of the shape of the underlying population if the sample size is sufficiently large.

The Central Limit Theorem can also be stated as follows:

- As $n \to \infty$, the the distribution of the sample mean becomes normal with mean μ , and standard deviation $\frac{\sigma}{\sqrt{n}}$

$$\lim_{n\to\infty} \bar{X} \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right)$$

- As $n \to \infty$, the distribution of the sample sum becomes normal with mean $n\mu$, and standard deviation $\sqrt{n}\,\sigma$

$$\lim_{n\to\infty} \sum_{i=1}^{n} X_i \sim N\left(n\mu, \sqrt{n}\,\sigma\right)$$

Probability and Statistics

- **Central Limit Theorem**
- **Demonstration with coins**

```
 ncoin = [1 3 5 10 20 50];
nroll=10000; % number of rolls
for i=1:length(ncoin),
ni=ncoin(i);
x=randi([0,1],ni,nroll) ; % coin flip: Head = 1 Tail =0
y=sum(x,1); % sample sum.
edges=min(y):max(y);
af=histc(y,edges); %absolute
rf=af/nroll; % relative
% plot figure
subplot(3,2,i)
stem(edges,rf,'filled')
title(['Number of Coins: n = ',num2str(ni)]);
xlabel('sample sum');
ylabel('rel. freq.');
end
```

Ordinary Least Square

Suppose we think there is a relationship

$$y = \alpha + \beta x$$

The Ordinarly Least Square (OLS) problem

$$\min_{\hat{\alpha},\hat{\beta}} \sum_{i=1}^{N} \left(y_i - \hat{\alpha} - \hat{\beta} x_i\right)^2$$

First order conditions (FOCs) are

$$\begin{aligned} \sum_{i=1}^{N} \left(y_i - \hat{\alpha} - \hat{\beta} x_i\right) &= 0 \\ \sum_{i=1}^{N} \left(y_i - \hat{\alpha} - \hat{\beta} x_i\right) x_i &= 0 \end{aligned}$$

Ordinary Least Square

Basic econometrics: the solution is

$$\begin{aligned}\hat{\beta} &= \frac{\sum_{i=1}^{N}\left(y_i-\bar{y}\right)\left(x_i-\bar{x}\right)}{\sum_{i=1}^{N}\left(x_i-\bar{x}\right)^2}\\ \hat{\alpha} &= \bar{y}-\hat{\beta}\bar{x}\end{aligned}$$

where

$$\begin{aligned}\bar{x} &= \frac{1}{N}\sum_{i=1}^{N}x_i\\ \bar{y} &= \frac{1}{N}\sum_{i=1}^{N}y_i\end{aligned}$$

Ordinary Least Square

We can write the original problem

$$\min_{\hat{\alpha},\hat{\beta}} \sum_{i=1}^{N} \left(y_i - \hat{\alpha} - \hat{\beta} x_i\right)^2$$

in matrix form

$$\min_b (y - Xb)' (y - Xb)$$

with solution

$$b = \left(X'X\right)^{-1} X'y$$

where

$$\begin{aligned} y &= \begin{bmatrix} y_1, & y_2, & \ldots, & y_N \end{bmatrix}' \\ X &= \begin{bmatrix} 1, & 1, & \ldots, & 1 \\ x_1, & x_2, & \ldots, & x_N \end{bmatrix}' \end{aligned}$$

and

$$b = \begin{bmatrix} \hat{\alpha}, & \hat{\beta} \end{bmatrix}'$$

Ordinary Least Square

Solve in different ways, but we need data. We generate artificial economic data

```
function GenerateData(alpha,beta,N)
x=[1:1:N]';
y=alpha + beta * x + randn(N,1);
save data x y;
figure(1);
plot(x,y,'b',x,alpha+x*beta,':r')
```

Ordinary Least Square

Use Matrix notation

```
function [alphaHat, betaHat] = OLS1
load data;
N = length(x)
X = [ones(N,1) x];
b = X\y;
%b = (X'*X)\(X'*y);
%b = inv(X'*X)*(X'*y);
alphaHat = b(1);
betaHat = b(2);
```

Ordinary Least Square

Use summations and means

```
function [alphaHat, betaHat] = OLS2
load data;
N = length(x);
meanX = sum(x)/N;
meanY = sum(y)/N;
betaHat = sum((x-meanX).*(y-meanY))./sum((x-meanX).^2);
alphaHat = mean(y) - betaHat * mean(x);
```

Ordinary Least Square

We derived FOCs, and then found an analytical solution to the system of two equations. Now we solve it numerically
We need a function which defines the system of FOCs:

```
function out = focs(in)
global x y
alphaHat = in(1);
betaHat = in(2);
out(1) = sum(y-alphaHat-betaHat.*x);
out(2) = sum((y-alphaHat-betaHat.*x).*x);
```

Ordinary Least Square

So, we find solution to the system of FOCs

```
function [alphaHat, betaHat] = OLS3
global x y
load data;
[sol, fval] = fsolve('focs',[2 1], optimset('TolFun',1.e-10));
error = sum(fval.^2);
if (error > sqrt(1.e-10))
disp('sol is not a zero!');
end
alphaHat = sol(1);
betaHat = sol(2);
```

Ordinary Least Square

Finally, we solve the original problem: minimisation of quadratic objective SSE (sum of squared errors)

$$\min_{\hat{\alpha},\hat{\beta}} \sum_{i=1}^{N} \left(y_i - \hat{\alpha} - \hat{\beta} x_i\right)^2$$

```
function out = SSE(in)
global x y
alphaHat = in(1);
betaHat = in(2);
out = sum( ( y- alphaHat - betaHat .* x ).^2 );
```

Ordinary Least Square

```
function [alphaHat, betaHat] = OLS4
global x y
load data;
[sol, fval] = fminsearch('SSE',[1 .5],
optimset('TolFun',1.e-15,'display','final'));
alphaHat = sol(1);
betaHat = sol(2);
```

Ordinary Least Square

Main file: calls all of them in turn

```
alpha = 0.5;
beta = 1.3;
N = 100;
GenerateData(alpha,beta,N);
[alphaHat1, betaHat1] = OLS1
[alphaHat2, betaHat2] = OLS2
[alphaHat3, betaHat3] = OLS3
[alphaHat4, betaHat4] = OLS4
```

Graphics in MATLAB

We have used

```
alpha = 0.5; beta = 2; N = 10;
x=[1:1:N]';
y=alpha + beta * x + randn(N,1);
save data x y;
figure(1);
plot(x,y,'b',x,alpha+x*beta,':r')
plot(x,y,'b',x,alpha+x*beta,':r','LineWidth',3) % width of line
```

Graphics in MATLAB

`hold on`

keeps the figure and allows to plot over existing lines:

```
plot(x,y,'b','LineWidth',2)
hold on;
plot(x,alpha+x*beta,':r','LineWidth',1)
hold off;
legend('data','fitted data');
xlabel('x'); ylabel('y');
text(5, 1,'add text here')
title('Ordinary Least Square Regression')
```

Graphics in MATLAB

```
figure(2)
subplot(1,2,1); plot(x,y,'b',x,alpha+x*beta,':r','LineWidth',3)
subplot(1,2,2); plot(x,y,'b',x,alpha+x*beta,':r')
figure(3)
subplot(2,2,1); plot(x,y,'b',x,alpha+x*beta,':r','LineWidth',3)
subplot(2,2,2); plot(x,y,'b',x,alpha+x*beta,':r')
subplot(2,2,3); plot(x,y,'b--',x,alpha+x*beta,':m')
subplot(2,2,4); plot(x,y,'g:',x,alpha+x*beta,'y-.','LineWidth',6)
```

Graphics in MATLAB

```
phi = 3;
c = [0.1:0.1:5];
n = [0.0:0.1:1];
[C,N] = meshgrid(c,n);
U = log(C) - N.^(1+phi)/(1+phi);
figure(4)
subplot(1,2,1); surf(C,N,U);
colormap('HSV');
xlabel('consumption'); ylabel('labour'); zlabel('utility')
subplot(1,2,2)
contour(C,N,U,'ShowText','on'); xlabel('consumption');
ylabel('labour');
```