Getting Started on Matlab
Econ 180.636

Matlab is a computer program for numerical computation. The variables in Matlab are scalars, vectors, matrices or higher-dimensional arrays. The name comes from MATrix LABoratory. These notes are intended to help you get started in Matlab and can be supplemented with the Matlab help facility and the manuals. Matlab is on all the computers in the graduate student lab

When you start up Matlab, you can work on it either interactively (typing commands into the command window), or you can save a file of commands to run. A program file ends in a .m extension. However, you should always work in programs to keep track of what you have been doing. When you want to run the program, you just type the program name at the Matlab command prompt without the .m suffix.

After each line, you should put a semicolon, unless you want the expression created by the line to be displayed on the screen.

If you put a "%" symbol, everything after that will be ignored—this is useful for commenting.

Reading in data

Matrices can be entered as

```
A=[1 2; 3  4];
```

which reads in the matrix $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$.

In the same way, you can enter a vector or a scalar. For example,

```
x=[1;2;3]  and x=[1 2 3]
```

enter column and row vectors respectively.

Basic matrix algebra

To refer to a particular element of a matrix, use the notation (i,j). For example A(1,1) is the 1,1 element of the matrix A.

To refer to columns or rows of a matrix, use the colon operator. For example A(:,1) is the first column of the matrix A. A(1,:) is the first row. A(:,[1 2]) is the first two columns.

The operators "+" "-", "*" and "^" refer to matrix addition, subtraction, multiplication and powers, respectively. For example C=A*B creates a matrix C that is A times B.

The operators ".*", "./" and ".^" refer to element by element multiplication, division and raising to a power. For example, if x=[1;2;3] and y=[4;5;6] then x.*y would be [4;10;18].

Defining a vector x=[a:b:c] gives a sequence starting at a and going up to c in increments of b, in a row vector. For example x=[1:2:7] would be the row vector with elements 1, 3, 5 and 7.

Dropping, saving and loading variables

To remove a variable (say x) type clear x

Anything you save will be saved to the current directory. Anything you load must be from the current directory. To change directory, type cd directory name. The current directory is displayed at the top of the screen.

To save the workspace, type save mydata. This saves your work in Matlab format in a file called mydata.mat. It can then be loaded with load mydata.

It is useful to be able to read and write data to/from an Excel spreadsheet.
m=xlsread('myspreadheet.xls','Sheet1','A1:B10');
reads the data in Sheet 1 of myspreadsheet.xls and puts it in a 10x2 matrix, m.

To put a Matlab variable into an Excel spreadsheet, type
xlswrite('myspreadsheet.xls',m,'Sheet1','A1:B2');

Useful operators

● x=sum(y)	If y is a vector, add up the elements. If y is a matrix, add up the columns
● x=mean(y)	As for sum, but averages the elements.
● x=eye(n)	Creates an nxn identity matrix
● x=zeros(n,m)	Creates an nxm matrix of zeros.
● x=ones(n,m)	Creates an nxm matrix of ones.
● x=rand(n,m)	Creates an nxm matrix of independent U[0,1] random variables.
● x=randn(n,m)	Creates an nxm matrix of independent standard normal random variables.
● A=inv(B)	Lets A be the inverse of the matrix B.
● A=B'	Lets A be the transpose of B.
● x=log(y)	Natural log
● x=exp(y)	Exponent
● x=diag(y)	If y is a vector, this reates a diagonal matrix with y on the diagonal If y is a matrix, it puts the diagonal of the matrix in the vector x.
● y=kron(A,B)	Lets y be the Kronecker product of A and B.
● y=sort(x)	Sorts the elements of the vector x in ascending order
● disp(x)	Displays x on the screen
● ceil(x)	Rounds x up to the nearest integer
● floor(x)	Rounds x down to the nearest integer
● round(x)	Rounds x off to the nearest integer

Hint: Each time you run the random number generator, it will give you different numbers. It may be helpful to always have the same seed. If you include the line

```
rand('seed',123)
```

(or some other number) then rand will always give the same random numbers. Similarly for randn. Random number generation is very important for the exercises in this course.

Loops and if statements

As an example of a loop in Matlab, suppose that A is a 5x5 matrix, then the program

```
for i=1:5;
        x(i)=sum(A(:,i));
end;
```

will loop over the columns of A and add them all up. This is a very inefficient way of doing the job however. You would get the same answer with x=sum(A)

As an example of an if statement, suppose that mark is a vector that denotes the mark that each of 30 students has got on a test. The commands

```
for i=1:30
        if mark(i)>0.6; pass(i)=1; else; pass(i)=0; end;
end; end;
```

creates a new vector with 1 for which students have passed and 0 for those that have failed.

A more efficient way of doing this is pass=mark>0.6;
So, in the same way, y=rand(100,1)>0.5 creates 100 Bernoulli random variables.

The find function

The find function is useful. It gives the indices of the elements of an array satisfying any given condition. For example x=ceil(6*rand(100,1)) generates 100 artificial dice rolls. Then y=find(x>=3) gives the indices of which of these were in fact 3 or higher.

Matlab graphics

If you have two vectors X and Y, to do a 2-dimensional plot of Y against X, type plot(y,x);

You can add in various options. For example

- plot(y,x,'LineWidth',2) makes the width of each line 2.
- plot(y,x,'r-') uses a red solid line
- axis([xmin xmax ymin ymax]) makes the X axis go from xmin to xmax and the Y axis go from ymin to ymax.
- xlabel('string') labels the horizontal axis. Similarly ylabel.
- title(' string ') puts a title over the whole picture
- text(x,y,' string') puts text at a given location
- gtext('string') lets the user select where to put the text (click on where the text is to go)

● line$([x_1 \ \ x_2],[y_1 \ \ y_2])$ draws a line from (x_1, y_1) to (x_2, y_2)
● subplot(n,m,i) splits the screen into an nxm matrix of plots and draws plot i.

Here is an example that illustrates some of these ideas

```
x=[0.1:0.1:1]';
subplot(2,2,1);
plot(x,log(x))
title('Log');
subplot(2,2,2);
plot(x,x.^2);
title('Quadratic');
subplot(2,2,3);
plot(x,exp(x));
title('Exponential');
subplot(2,2,4);
plot(x,2*x);
title('Linear');
axis([0 1 0 3]);
```

You can save pictures in pdf files or copy them into Word.

A histogram is a useful plotting tool in Matlab.

```
hist(y,b)
```

creates a histogram with b equally sized bins. Meanwhile hist(y) uses the default number of bins.

As an example,   y=rand(1000000,1) creates 1,000,000 uniform random numbers. Then

```
hist(y,10)
hist(y,20)
hist(y,50)
```

creates histograms with successively smaller bin sizes. Intutively, this is getting closer and closer to the probability density function.

Matlab functions

Functions are m-files that can accept input arguments and return output arguments. The names of the m-file and of the function should be the same. Matlab has its own built in functions (we've seen several of them), but you can write your own too.

Here is a simple illustrative function

```
function y=myfunc(x);
y=x^2;
```

You then save this as myfunc.m. If you are on this directory and type y=myfunc(2), then it will set y=4.

An illustrative program

We can put this together with a little demonstration of the Bayes rule “game show” problem that we discussed in class. Call door 1 the door with the prize. Now suppose that the contestant picks a random door, x. The host then opens one of the doors other than this door and other than door 1, that does *not* contain the prize. Call this door y. The contestant can switch door or not. Here is how we can simulate the probability of winning the prize if the contestant switches and if she does not artificially repeating the game 10,000 times (a simple Monte-Carlo experiment).

```
rand('seed',123);               %So as to get the same answer each time program runs
n=10000;                        %The number of simulations
x=ceil(rand(n,1)*3);            %The door chosen by the contestant (1,2or 3)
for i=1:n;                      %y is the door opened by the host
        if x(i)==1; y(i,1)=1+ceil(rand(1,1)*2); end;
        if x(i)==2; y(i,1)=3; end;
        if x(i)==3; y(i,1)=2; end;
end;
z=6-x-y;                        %The door chosen if the contestant switches
mean(x==1)                      %Proportion of times wins if doesn't switch
mean(z==1)                      %Proportion of times wins if does switch
```

Running this program gives the answers 0.3318 and 0.6682, very close to the theoretical values of 1/3 and 2/3 respectively.

The loop in the middle could be avoided easily by typing in
y=((1+ceil(2*rand(n,1))).*(x==1))+(3*(x==2))+(2*(x==3))
As you get more experience in Matlab, you should try to avoid loops, as it slows Matlab down considerably.

Matlab toolboxes

There are a number of matlab “toolboxes” that give additional functions, including numerical optimization, finance and statistics toolboxes. The statistics toolbox gives random numbers from different distributions, and cumulative distribution functions and probability density functions. In most cases, you should be able to construct these without using the statistics toolbox and that is important for learning and will generally be a requirement for homework. Still you can see the list of statistics toolbox functions if you go to

http://www.mathworks.com/products/statistics/

and click on function list.