1 MATLAB Operations

Matlab uses the symbols *,/,+,-,and ˆ to denote multiplication, division, addition, subtraction and exponention, respectively. For the case of matrix multiplication, * will multiply the matrices as we typically do. For element-by-element calculations, you need to insert a "." in front of the operation. For example, x.*y will compute the element-by-element product of vectors x and y of equal lengths.

2 Inputting Data

Suppose that we want to define a 3×1 vector x, with elements given as 3,5 and 1. We do this as follows:

$$x = [3\ 5\ 1]'$$

(The ' makes it a column vector, and [and] denote the beginning and end of a matrix or vector, respectively.)

Alternatively, we could write

$$x = [3; 5; 1]$$

where the ; defines a new row.

Suppose we wanted to enter the following matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

We would do this in MATLAB by typing

$$A = [1\ 2\ 3;\ 4\ 5\ 6;\ 7\ 8\ 9],$$

for example.

Note: If you include a ; at the end of the statement, MATLAB will not display the output. Otherwise, it will display the output.

Exercise 1: Calculate the following A*B, A.*B, A*x, x*A.

Exercise 2: Plotting a Standard Normal Density and using **trapz** command

```
xgrid = linspace(-3.5,3.5,100);
density = (1/sqrt(2*pi))*exp(-.5*xgrid.^ 2);
plot(xgrid,density);
trapz(xgrid,density)
trapz(xgrid,(xgrid.*density))
trapz(xgrid,( (xgrid.^ 2).*density))
```

3 Commenting

Matlab Uses the percentage sign to denote a comment. It will ignore everything appearing after the percentage sign, so this provides an opportunity for you to add your own comments to your code.

4 Useful Matlab Commands

- **inv(x)** Inverse of a matrix x.
- **det(x)** Determinant of matrix x
- **rand(n,k)** An $n \times k$ matrix of numbers uniformly distributed on [0,1].
- **randn(n,k)** An $n \times k$ matrix of numbers drawn from the standard Normal distribution.

- **clear** Clear MATLAB's memory of all variables
- **clc** Clear MATLAB's screen
- **zeros(n,k)** Create an $n \times k$ matrix of zeros
- **ones(n,k)** Create and $n \times k$ matrix of ones
- **eye(k)** Create a $k \times k$ identity matrix
- **length(k)** The length of a vector or matrix x. In the case of a matrix, the length is defined to be the maximum of the column and row dimensions.
- **size(k)** The size (number of rows and columns) of a matrix x. [i.e., [n,k]= size(x)]
- **find** Find elements of a vector (matrix) with certain value [i.e., points = find(x==1)]
- **resphape** Reshape a given matrix or vector.
- **blockdiag** Create a block diagonal matrix
- **diag** Take the diagonal elements of a square matrix.
- **min, max, std, var, mean** Compute minimum, maximum, variance, mean of a vector (or matrix). For the case of a matrix, these calculate the statistic for each column of the matrix.
- **save** Save a data set in MATLAB format.
- **print** Print a MATLAB-produced figure.
- **sortrows** Sort a given matrix of data by a particular column.
- **disp** Add text with your output [i.e., disp('Here are the OLS results')]

For any matlab command, typing **help** followed by the command name will give you information and options regarding that command.

5 Looping

```
iter = 5;
x = zeros(iter,1);
for i = 1:iter;
x(i,1) = 5*i;
end;
```

```
randn('seed',sum(100*clock));
iter1 = 5;
iter2 = 10;
x = zeros(iter2,iter1);
for i = 1:iter1;
tempp = randn(iter2,1);
x(:,i) = tempp.^2;
end;
```

6 Interactive versus batch mode

Interactive mode refers to performing a set of operations directly in MATLAB's command window. This is not generally a good idea, particularly if you need to write a complex program. More generally, you can write an entire set of code in an editor, save it, and then execute it within matlab. This makes it much easier to fix bugs, as you won't have to re-type the whole thing again.

Along this lines, you can create and save files that perform specific functions, and then call upon these files later to do the same job.

Exercise 3: Let us first create an m-file that calculates the OLS estimates.

```
function [bhat] = myols(x,y);
bhat = inv(x'*x)*x'*y;
```

Save this m-file as myols.

Now, load the data on the course website into MATLAB and calculate the OLS estimates;

```
load labsupply.txt;
weeks = labsupply(:,1);
afqt = labsupply(:,4);
spouseinc = labsupply(:,5);
kids = labsupply(:,6);
education = labsupply(:,7);
nobs = length(weeks);
x = [ones(nobs,1) afqt spouseinc kids eduction];
bhat = myols(x,weeks)
```

Updating the myols m-file

```
function [bhat,tstat] = myols(x,y);
bhat = inv(x'*x)*x'*y;
nobs = length(y); sigsq = (1/nobs)*(y-x*bhat)'*(y-x*bhat);
varcov = sigsq*inv(x'*x);
stddevs = sqrt(diag(varcov));
tstat = bhat./stddevs;
```