UNIVERSITY OF CHICAGO
Booth School of Business

Bus 35120 – Portfolio Management Prof. Lubos Pastor

Introduction to MATLAB

The purpose of this note is to help you get acquainted with the basics of Matlab. Matlab is used in several of our advanced finance courses. It is powerful, simple, and user-friendly.

1 Getting Help

Matlab has an extensive and amazingly good help system. For a quick overview, type `helpdesk` at the command line. (The command line begins with ">>").

If you need help with a specific command `command1`, type `help command1`. For example, `help sum` will display the help file for the `sum` command.

If you do not know the name of a command but you know what it is used for, you can search for the command using `lookfor`. Typing `lookfor word1` will scan all help files and select those that contain the word `word1`. For example, to find out how to estimate variance, type `lookfor variance` and one of the top "finds" will be the `var` command you need. (If the `lookfor` command is running too long for your taste, interrupt it by pressing Ctrl-C.)

Do not feel overwhelmed by the number of various Matlab commands that are available. You can get by easily (in this course, anyway) with just a few basic commands that will quickly emerge in various hints that you will receive. (I feel the same way about English: some dictionaries have tens of thousands of English words, of which I know only a fraction.) By the end of the quarter, you will consider Matlab to be your friend.

2 Basic Algebra

Matlab can serve as a sophisticated calculator. At the command line, type

```
2+2
```

and press enter (you'll get "4", of course). Other symbols you should get used to include `-`, `*`, `/`, and `^`. The operations they correspond to are subtraction, multiplication, division, and raising to a power (for instance, `2^3` will raise 2 to the third power).

Moreover, Matlab will work with parentheses: "(" and ")." Try typing:

```
(2*(1+3)+2)^5
```

Rather than input raw numbers into Matlab, you can define constants and give them values. For instance, `x=2` will create variable `x` and assign it the value of 2. Then you can write `x^2+1`, or, if we define another variable `y`, we could try `x/y+2*x*y` etc.

So far, we have only worked with numbers (scalars). The big strength of Matlab is manipulating vectors and matrices (or tables of numbers). As an example, type

```
v1=[1;2;3]
```

and press enter. You just created a column vector `v1`. To create a similar row vector, type `v2=[1,2,3]`. Alternatively, try `v2=v1'`. Putting an apostrophe after the variable transposes it (flips it on its side).

The cells of the vector are indexed. You can see a particular element by typing its index number, e.g., `v1(2)` will yield the second element of the vector. To get more than one element, we use the ":" symbol. Typing `v1(2:3)` returns the second and third elements of `v1`. A useful alternative is typing `v1(2:end)` , which gives the elements starting at the second one and ending at the last one. What does `v1(2:end-1)` give you?

Now define another vector, `v3=[0;1;3]`, and suppose you want to multiply all elements of `v1` by the corresponding elements of `v3`. To do that, type

```
v1.*v3
```

The dot before * tells Matlab to multiply element by element. You can only do that if both vectors are of the same size (otherwise you get an error message; try `v1.*v2`).

Matrices are lists of vectors (or tables of numbers). Let's define a matrix

```
m1=[1,2;3,4;5,6]
```

You can call its elements by invoking their two coordinates. For instance, try typing `m1(1,2)`, `m1(3,end)`, or `m1(end,2)`. The first coordinate points to a row, the second one: to a column. You can also ask Matlab for the whole row (try `m1(2,:)`) or for the whole column (try `m1(:,2)`). To see the size of the matrix, type `size(m1)`. The result is a matrix, and we can define a new variable to store it:

```
m_size=size(m1)
```

You can further define new variables for the number of rows and columns by typing:

```
nr_rows=m_size(1)
nr_cols=m_size(2)
```

To see the list of all variables that are currently defined, along with their dimensions, type `whos`. This is useful if you work with several variables at the same time.

There are many other fun things to do with vectors and matrices, but these are the most important ones. Experiment a bit to get some practice!

3 Writing a Program

When you input commands at the command line, as we have done so far, Matlab executes them one by one. For larger jobs, it is more convenient to write a set of commands (a program) using Matlab's editor. To start the editor, type `edit`.

In the text of the program, you can type commands just like you did before. Once you are done, click "File → Save" (or the diskette icon), select the name for your program (e.g., "simulation") and save it on your disk. Matlab will automatically give the program its proper extension ".m" (so your program file is called "simulation.m").

You can download a sample Matlab program, called "simulation.m," from Chalk. This program may be useful for Assignment #2.

You can run the program by typing its name at the Matlab command line (e.g., `simulation`). (Note: If the program runs too long for your taste, you can interrupt it by pressing Ctrl-C.)

If you chose to save the program in a different directory than the current one, Matlab will not find it and won't be able to execute it. To find the name of the current directory, type `cd` . To change to a different directory, type `cd new_directory_name` . Typing `cd ..` moves you one level up in the directory tree.

It is a good idea to establish a dedicated directory for this course on your hard drive. For example, you can create a directory *My Documents/Booth/PfMgt/MATLAB*, and do all your MATLAB-related work in that directory. You can download all data from Chalk into that directory. When you open MATLAB, use `cd` to change your directory into *My Documents/Booth/PfMgt/MATLAB*, and once you're there, you can do all your work.

To see what files are available in the current directory, type `dir` at the command line. If the datafile does not show up, it is not there, and you won't be able to load it until you copy the file into that directory. Similarly, if a program such as *simulation.m* does not show up when you type `dir`, it won't run; first, you'll have to download it from Chalk into that directory. Again, make sure that the data you want to load as well as any programs you want to run are in your current directory.

After Matlab executes a command, it displays the result immediately. This can be annoying if you are running a program, with too many results printed out on the screen. To stop it from displaying the result, put ";" at the end of the command. For instance, typing `m1=[1,2;3,4;5,6];` will define matrix `m1` as before, but will not print it on the screen. To see it, type `disp(m1)`, or simply the name of the matrix, `m1`.

If you want Matlab to do the same thing many times over, it will gladly do so. One way is to use a `for` loop. Type `for i=1:5 a(i)=i; end`, and you will have created a vector a containing numbers 1, 2, 3, 4, and 5. (An easier way is to type `a=1:5`, but I am just illustrating a loop.) Another example of a loop is in the sample program simulation.m.

Finally, it is a good policy to clean the workplace before you start with a program. To do that, type `clear all`. This command will erase all variables you've defined so far. You can also type `close all`, which will close all auxiliary windows you opened during your previous work with Matlab. You can add these two commands at the beginning of all your programs.

4 Working with Data

All data in this class will be given to you in text files. You can open them with a Wordpad or your favorite text editor to see what they look like inside.

To load a file into Matlab, use the `load` command. Suppose your file name is `returns_annual.txt`. You can define a new variable, `x`, and set it equal to the contents of the file by typing

```
x=load('returns_annual.txt');
```

Note that ";" at the end of the command will stop Matlab from displaying the contents of the file on the screen.

You can also simply type `load returns_annual.txt`, hit enter, and then type `whos` to see what variables you have loaded in.

Data will always be a matrix. You can see its contents by typing `disp(x)`. Moreover, you can define a new vector equal to, say, the second column of `x` by typing `Rstock=x(:,2)`.

Let us summarize the most important properties of `Rstock`. You can get its average, variance, and standard deviation by typing

```
mean(Rstock)

var(Rstock)

std(Rstock)
```

You can also get the means etc of all columns of `x` at the same time by typing `mean(x)` or `std(x)`. These commands are among those you will need most often in this course.

To get the matrix of covariances between the columns of the data matrix, type `cov(x)`. Typing `corrcoef(x)` yields correlations. To get the correlations only for two variables (say, the second and the third columns of `x`), use `corrcoef(x(:,2),x(:,3))`.

5 Simulating Random Variables

When you type `randn` at the command line and hit enter, Matlab will generate a random variable from the standard normal (N(0,1)) distribution. If you want Matlab to generate a

10×1 column vector of such variables, type `randn(10,1)`. If you want a 10×5 matrix of random N(0,1) variables, type `randn(10,5)`.

To simulate a random variable y with mean m and variance v, write `y = m + sqrt(v)*randn`. To simulate a 10×5 matrix of such variables, type `y = m + sqrt(v)*randn(10,5)`.

To generate random variables from a standard uniform distribution, type `rand`.

One way to draw a random integer y from the set $\{1, 2, \ldots, N\}$ (which can be useful in resampling methods) is the following: `y = ceil(N*rand)`. The `ceil` command rounds a real number up to the nearest integer value (e.g., `ceil(1.2)=2`). Similarly, `floor` rounds down (e.g., `floor(1.2)=1`).

6 Producing Plots

To do a time series plot of your variable (say, `Rstock`), you can simply type `plot(Rstock)`. To plot the returns against the time index, type `plot(x(:,1),Rstock)`.

You can "beautify" the graph using `title()`, `xlabel()`, and `ylabel()`. Try

```
title('My first graph')

xlabel('year')

ylabel('return')
```

The other kind of plot that you may need is a histogram. To get the histogram of `Rstock`, type `hist(Rstock)`. Matlab will select the number of bins by default, although you can force it to use a particular number (say, 23 bins) by typing `hist(Rstock, 23)`.

To plot the normal curve over your histogram, you can type `histfit(Rstock)`.

Finally, unless you open a new window, Matlab will erase the old graph before it plots a new one. To ensure this does not happen, type `figure` before you create a new picture.

We have covered quite a few commands this time. We will learn some new structures and some new commands in the coming weeks. To ensure that you quickly learn how to use them, play around with the ones summarized here and get used to them. After all, you will be using them again and again in this course.