MATLAB session

Introduction

To get the most out of this course, you will need to work through the notes while using MATLAB. The notes are accompanied by MATLAB m.files containing the codes and data files for this session's exercises. With some brief preliminaries, this introductory section illustrates how to set up MATLAB for the first time and refers to a podcast illustrating a brief overview of the visual layout of MATLAB desktop on the computer. The next section deals with some basic building blocks of MATLAB. Then, we will explore more advanced features such as loops, plots and user written functions. The course will closely follow Frain (2010a) [1] and his notes are more encompassing than these ones.

Preliminaries

MATLAB is short for Matrix Laboratory as it was originally designed to allow matrix multiplications and students access to the library for performing numerical linear algebra, LINPACK, without learning FORTRAN in the late 1970s by Cleve Moler at New-Mexico University. In the early 1980s, in tandem with Jack Little and Steve Bangert, MATLAB was commericialised and its programming was then written and compiled in C. Unlike more menu-driven econometric packages such as Microfit and EViews, MATLAB sometimes requires the user to do more work. However, seeing what goes on under the black box can be helpful for understanding methodology and allows greater control, especially in situations where we need to be creative in our programming (e.g. utilising new econometric methods not yet incorporated in software packages).

The main references for these notes are Frain (2010a) and Frain (2010b) [2] – they are highly recommended reading for MATLAB and Stata, respectively, having a more in depth and broad treatment of much of the theory in the first two sections of these present notes. They also contain an extensive review of alternatives plus ways to manage your data, create reports, etc. They are also required reading for today's session.

In organising your work, Frain (2010a) [1] suggests you do the following:

1. For each project, set up a new directory associated with that project (e.g. `C:\Matlab\project1`).
2. For each project, set up a shortcut for that project. In the shortcut, you should specify that MATLAB starts in the directory for that project. This can be accomplished as follows:
   - Open Windows Explorer and navigate to the directory containing the MATLAB.exe file; hold the right button on your mouse to drag it to your project directory and then select 'Create short cut here' from the menu that appears; if you already created a short cut to MATLAB in another project directory, then you can copy it to your new project directory.
   - In Windows Explorer, right click on the short cut, click on properties and then ensure the 'Start in' box refers to your new project directory, e.g. `C:\Matlab\project1`.
3. Change the options in Windows explorer so full filenames (i.e. including the file extension) appear when you search for files; the default setup in MS Windows is not to show file extensions. This can be accomplished as follows: open Windows Explorer, select `Organize -> Folder and search options` and select the View tab; then ensure that the box saying 'Hide extensions for known file types' is unticked; finally, click Apply and click 'Apply to Folders' to reset all folders. See Frain (2010b) [2] for a further discussion on this. Also see `podcast1a` available at www.michael-curran.com

MATLAB Desktop

See `podcast1b` available at www.michael-curran.com

Basics

Now having presented the rudimentary organisation of MATLAB, we will look at a first sample session to develop the basic concepts and tools you will need for utilising MATLAB. In particular, given how MATLAB is structured around a matrix-environment language, we will look at manipulations involving matrices first before moving onto reading data into MATLAB and exporting output from MATLAB.

Matrices

See the sample MATLAB session file: `matrices.m` in conjunction with Frain (2010a)[1] section 2.

Data

See `podcast2` available at www.michael-curran.com in conjunction with Frain (2010a)[1] section 3.

Advanced

Having looked at basic matrix manipulation and data input/output, we now move onto more advanced topics. First we will look at the construction of loops and then explore the creation of MATLAB graphics. Since MATLAB allows us to write scripts (m files with a list of instructions), we will investigate these further. Many toolboxes are available in MATLAB (e.g. Statistics, Optimisation, Parallel Programming, etc.) and a few will be briefly mentioned here. This section will close with a few comments on the availability of alternatives to MATLAB.

Loops

This subsection is based on Frain (2010a)[1] section 4. See the sample MATLAB session file `loopplotfunc.m` available at www.michael-curran.com.

The four basic loops are:

1. **if**
2. **for**
3. **while**
4. **switch**

The syntax for **if** statements is as follows:

```
if conditions
  statements
end
```

where conditions may include operators such as those in table 1

==	equal	&&	logical and (for scalars) short-circuiting
∼=	not equal	|	logical or
<	less than	||	logical or and (for scalars) short-circuiting
>	greater than	xor	logical exclusive or
<=	less than or equal to	all	true if all elements of vector are nonzero
&	logical and	any	true if any element of vector is nonzero

Table 1: Operators.

If the conditions are true, then the statements are processed. The **if** statement can be extended to process different statements depending on whether the conditions are true:

```
if conditions
  statements1
else
  statements2
end
```

Here statments1 are processed if conditions are true and statements2 are processed if conditions are false. Finally, you can add extra sets of conditions with the **elseif** statement as follows:

```
if conditions1
  statements1
```

```
elseif conditions2
  statements2
else
  statements3
end
```

The structure of **for** loops is as follows:

```
for variable = expression
  statements
end
```

The structure of **while** loops is as follows:

```
while conditions
  statements
end
```

An advantage of the **while** loop over the **for** loop is that you do not need to know as much information (**for** requires you to know exactly when and for how many occurences an operation will be repeated).

The structure of the **switch** loop is as follows:

```
switch a
  case 1
    x = 10
  case 2
    x = 15
  case 3
    x = 5
  otherwise
    error('a must be 1, 2 or 3')
end
```

Remark. You can use loops within loops too. When using loops to fill the elements of a vector or a matrix, you should initialise the vector or matrix prior to doing so. However, matrix statements are more efficient and easier to use than loops, so they are to be preferred, unless you must use loops.

Plots

See the MATLAB sample session `loopplotfunc.m` file.

User written functions

You may write your own functions and use them in the same manner you would use a pre-written MATLAB function.[1] See the example given with explanation on how to do this in section 7 of Frain (2010a) [1]. Also see the MATLAB sample session `myfunction.m` called from `loopplotfunc.m` available at www.michael-curran.com.

[1] Where m.files contain lists of instructions (i.e. script files), we are able to repeat analyses without having to retype all the instructions, open datasets, etc. In effect, we can automate much of the work including documentation such as reporting and publishing. The publish command is very useful and so is offloading through batch files.

Toolboxes

There are many MATLAB toolboxes, which are usually extra add-ons from the basic price. Among others, they include the following.

Curve Fitting Toolbox provides graphical tools and functions for fitting curves and surfaces to data. Datafeed Toolbox provides data from data providers into MATLAB through a series of function calls. Econometric Toolbox has functions for modelling economic data. Financial Toolbox has functions for constructing mathematical models with financial data and performing statistical analysis of financial data. Financial Derivatives Toolbox contains functions for analysing equity and fixed-income derivatives. Fixed-Income Toolbox provides functions that can be used for fixed-income analysis and modelling. Global Optimization Toolbox allows the user to search for global solutions to problems having multiple maxima or minima and nonsmooth optimization problems. Optimization Toolbox contains algorithms for standard and large-scale optimization problems. Parallel Computing Toolbox allows users to perform parallel computations on multicore processors, GPUs and clusters. Partial Differential Equation Toolbox includes tools allowing the user to study and solve partial differential equations in two-space dimensions and time (using finite element methods). Statistics Toolbox provides many algorithms and tools for organizing, analyzing and modelling data. Symbolic Math Toolbox provides tools allowing the user to solve symbolic mathematical expressions and perform variable-precision arithmetic; see also MuPAD.

You can visit www.mathworks.co.uk/help for more information on all toolboxes.[2]

Alternatives

While EViews has also traditionally been used in many masters programs for the time-series component of econometrics, programs like Gauss, Mathematica, Octave, Scilab and R are closer in nature to MATLAB. Octave is an open-source program that is very similar to MATLAB and discussed by Frain (2010a)[1] along with Scilab and R in section 10. Without having access to MATLAB, one can download Octave, Scilab and R for free. The first two are very similiar to MATLAB and so learning these can be useful if one does not have access to MATLAB. My personal advice would be to use Octave if you don't have MATLAB, but if you really want to use open-source programs and you have the patience to work through the 'steep learning curve', R is a very well worthwhile program to learn having a huge amount of non-commercial support in terms of community groups. For more on free software for econometrics and economics, see John Frain's section of his website www.tcd.ie/Economics/staff/frainj/home.htm on Free software for Econometrics: www.tcd.ie/Economics/staff/frainj/main/freeSoftware/freeSoftware.html.

[2] Frain (2010a) [1] provides a very good first treatment of the LeSage Econometric Toolbox in section 8 and in his appendix.

*

References

[1] Frain, J.C. (2010a) 'An introduction to Matlab for econometrics' *Trinity Economics Papers* tep0110, Trinity College Dublin, Department of Economics.

[2] Frain, J.C. (2010b) 'Introduction to Stata with econometrics in mind' *Trinity Economic Papers* tep0210, Trinity College Dublin, Department of Economics.