An Introduction to Matlab for Econometrics

John C. Frain

TEP Working Paper No. 0110

February 2010

Trinity Economics Papers

Department of Economics

Trinity College Dublin

An Introduction to MATLAB for Econometrics

John C. Frain. [*]

February 2010

Abstract

This paper is an introduction to MATLAB for econometrics. It describes the MATLAB Desktop, contains a sample MATLAB session showing elementary MATLAB operations, gives details of data input/output, decision and loop structures, elementary plots, describes the LeSage econometrics toolbox and maximum likelihood using the LeSage toolbox. Various worked examples of the use of MATLAB in econometrics are also given. After reading this document the reader should be able to make better use of the MATLAB on-line help and manuals.

Contents

[*]Comments are welcome. My email address is `frainj at tcd.ie`

1 Introduction

1.1 Preliminaries

These notes are a guide for students of econometrics who wish to learn MATLAB in MS Windows. I define the functions of MATLAB using simple examples. To get the best benefit from these notes you should read them sitting in front of a computer entering the various MATLAB instructions as you read the notes. The material in the first three sections is elementary and will be required by all economists starting with MATLAB. The remaining sections contain some more advanced material and should be read as required.

In these notes I have used a mono-spaced font for MATLAB instructions and for computer input and output. Descriptive material, explanations and commentary on the computer input/output is given in the current font.

While the first aim of these notes is to get the reader started in the use of MATLAB for econometrics it should be pointed out that MATLAB has many uses in economics. In recent years it has been used widely in what has become known as computational economics. This area has applications in macroeconomics, determination of optimal policies and in finance. Recent references include Kendrick et al. (2006), Marimon and Scott (1999), Miranda and Fackler (2002) and Ljungqvist and Sargent (2004).

I do not know of any book on MATLAB written specifically for economics. Creel (2008) is a set of lecture notes on econometrics which can be downloaded from the web. This contains examples of quantitative econometric analysis using GNU Octave which has a syntax similar to Matlab (see section 10.1). LeSage (1999) is a free econometrics toolbox available for download from `http://www.spatial-econometrics.com/`. This site also contains links to several other MATLAB resources useful in econometrics. A free ARCH/GARCH toolbox is available at `http://http://www.kevinsheppard.com/wiki/MFE_Toolbox`. MathWorks, the composers of MATLAB have a list of books using MATLAB for Economics/Finance (see `www.mathworks.com`). They have also issued a new econometrics toolbox (see `http://www.mathworks.com/products/econometrics/`). The MathWorks overview of this toolbox indicates that is is targeted at econometric time series in finance. For advanced applications in applied probability Paolella (2006, 2007) are comprehensive accounts of computational aspects of probability theory using MATLAB. Higham and Higham (2005) is a good book on MATLAB intended for all users of MATLAB. Pratap (2006) is a good general "getting started" book. There are also many excellent books covering MATLAB for Engineers and/or Scientists which you might find useful if you need to use MATLAB in greater depth.

These notes can not give a comprehensive account of MATLAB. Your copy of MATLAB comes with one of the best on-line help systems available. Full versions of the manuals are available in portable document format on the web at `http:/www.mathworks.com`.

MATLAB started life, in the late 70's, as a computer program for handling matrix operations. Over the years it has been extended and the basic version of MATLAB now contains more than 1000 functions. Various "toolboxes" have also been written to add specialist functions to MATLAB. Anyone can extend MATLAB by adding their own functions and/or toolboxes. Any glance at an econometrics textbook shows that econometrics involves much matrix manipulation and MATLAB provides an excellent platform for implementing the various textbook procedures and other state of the art estimators. Before you use MATLAB to implement procedures from your textbook you must understand the matrix manipulations that are involved in the procedure. When you implement them you will understand the procedure better. Using a black box package may, in some cases, be easier but how often do you know exactly what the black box is producing. Using MATLAB for econometrics may appear to involve a lot of extra work but many students have found that it helps their understanding of both matrix theory and econometrics.

In MATLAB as it all other packages it makes life much easier if you organise your work properly. The procedure That I use is some variation of the following –

1. Set up a new directory for each project (e.g. `s:\Matlab\project1`)
2. Set up a shortcut for each project. The shortcut should specify that the program start in the data directory for the project. If all your work is on the same PC the shortcut is best stored on the desktop. If you are working on a PC in a computer lab you will not be able to use the desktop properly and the shortcut may be stored in the directory that you have set up for the project. If you have several projects in hand you should set up separate shortcuts and directories for each of them. Each shortcut should be renamed so that you can associate it with the relevant project.
3. Before starting MATLAB you are strongly advised to amend the options in Windows explorer so that full filenames (including any file extensions allocated to programs) appear in Windows Explorer and any other Windows file access menus.

1.2 The MATLAB Desktop

The MATLAB desktop has the following parts -

1. The Command Window
2. The Command History Window
3. The Start Button
4. The Documents Window (including the Editor/(Debugger) and Array Editor
5. The Figure Windows
6. The Workspace Browser
7. The Help Browser
8. The Path Browser

1.2.1 The Command Window

The simplest use of the command window is as a calculator. With a little practice it may be as easy, if not easier, to use than a spreadsheet. Most calculations are entered almost exactly as one would write them.

```
>> 2+2
ans = 4

>> 3*2
ans = 6
```

The object **ans** contains the result of the last calculation of this kind. You may also create an object `a` which can hold the result of your calculation.

```
>> a=3^3
a = 27
>> a
a = 27

>> b=4^2+1
b = 17

>> b=4^2+1;

% continuation lines
>> 3+3 ...
+3
ans = 9
```

Type each instruction in the command window, press enter and watch the answer. Note

- The arithmetic symbols `+, -, *, /` and `^` have their usual meanings
- The assignment operator `=`
- the MATLAB command prompt `>>`
- A ; at the end of a command does not produce output but the assignment is made or the command is completed
- If a statement will not fit on one line and you wish to continue it to a second type an ellipsis (. . .) at the end of the line to be continued.

Individual instructions can be gathered together in an m-file and may be run together from that file (or script). An example of a simple m-file is given in the description of the Edit Debug window below. You may extend MATLAB by composing new MATLAB instructions using existing instructions gathered together in a script file.

You may use the up down arrow keys to recall previous commands (from the current or earlier sessions) to the Command Window. You may the edit the recalled command before running it. Further access to previous commands is available through the command window.

1.2.2 The Command History Window

If you now look at the Command History Window you will see that as each command was entered it was copied to the Command History Window. This contains all commands previously issued unless they are specifically deleted. To execute any command in the command history double click it with the left mouse button. To delete a commands from the history select them, right click the selection and select delete from the drop down menu.

1.2.3 The Start Button

This allows one to access various MATLAB functions and works in a manner similar to the Start button in Windows

1.2.4 The Edit Debug window

Clearly MATLAB would not be of much use if one was required, every time you used MATLAB, to enter your commands one by one in the Command Window. You can save your commands in an m-file and run the entire set of commands in the file. MATLAB also has facilities for changing the commands in the file, for deleting command or adding new commands to the file before running them. Set up and run the simple example below. We shall be using more elaborate examples later

The Edit Window may be used to setup and edit M-files. Use `File|New|m-file` to open a new m-file. Enter the following in the file `\vol\_sphere.m`

```
% vol_sphere.m
% John C Frain revised 12 November 2006
% This is a comment line
% This M-file calculates the volume of a sphere
echo off
r=2
volume = (4/3) * pi * r^3;
string=['The volume of a sphere of radius ' ...
    num2str(r) ' is ' num2str(volume)];
disp(string)
% change the value of r and run again
```

Now Use `File|Save As vol_sphere.m`. (This will be saved in your default directory if you have set up things properly check that this is working properly).

Now return to the Command Window and enter `vol_sphere`. If you have followed the instructions properly MATLAB will process this as if it were a MATLAB instruction.

The edit window is a programming text editor with various features colour coded. Comments are in green, variables and numbers in black, incomplete character strings in red and language key-words in blue. This colour coding helps to identify errors in a program.

The Edit window also provides debug features for use in finding errors and verifying programs. Additional features are available in the help files.

1.2.5 The Figure Windows

This is used to display graphics generated in MATLAB. Details will be given later when we are dealing with graphics.

1.2.6 The Workspace Browser

This is an option in the upper left hand window. Ensure that it is open by selecting the workspace tab. Compare this with the material in the command window. Note that it contains a list of the variables already defined. Double clicking on an item in the workspace browser allows one to give it a new value.

The contents of the workspace can also be listed by the `whos` command

1.2.7 The Help Browser

The Help Browser can be started by selecting the [?] icon from the desktop toolbar or by typing `helpdesk` or `helpwin` in the Command Window. You should look at the Overview and the Getting Help entries. There are also several demos which provide answers to many questions.

MATLAB also has various built-in demos. To run these type demo at the command prompt or select demos from the start button

The on-line documentation for MatLab is very good. The MatLab www site (`http://www.mathworks.com` gives access to various MatLab manuals in pdf format. These may be downloaded and printed if required. (Note that some of these documents are very large and in many cases the on-line help is more accessible and is clearer.

One can also type `help` at a command prompt to get a list of help topics in the Command Window. Then type `help topic` and details will be displayed in the command window.

If, for example, you wish to find a help file for the inverse of a matrix the command `help inverse` will not help as there is no function called inverse. In such a case one may enter `lookfor inverse` and the response will be

```
INVHILB Inverse Hilbert matrix.
IPERMUTE Inverse permute array dimensions.
ACOS   Inverse cosine, result in radians.
ACOSD  Inverse cosine, result in degrees.
ACOSH  Inverse hyperbolic cosine.
ACOT   Inverse cotangent, result in radian.
ACOTD  Inverse cotangent, result in degrees.
ACOTH  Inverse hyperbolic cotangent.
ACSC   Inverse cosecant, result in radian.
ACSCD  Inverse cosecant, result in degrees.
```

```
ACSCH  Inverse hyperbolic cosecant.
ASEC   Inverse secant, result in radians.
ASECD  Inverse secant, result in degrees.
ASECH  Inverse hyperbolic secant.
ASIN   Inverse sine, result in radians.
ASIND  Inverse sine, result in degrees.
ASINH  Inverse hyperbolic sine.
ATAN   Inverse tangent, result in radians.
ATAN2  Four quadrant inverse tangent.
ATAND  Inverse tangent, result in degrees.
ATANH  Inverse hyperbolic tangent.
ERFCINV Inverse complementary error function.
ERFINV Inverse error function.
INV    Matrix inverse.
PINV   Pseudoinverse.
IFFT Inverse discrete Fourier transform.
IFFT2 Two-dimensional inverse discrete Fourier transform.
IFFTN N-dimensional inverse discrete Fourier transform.
IFFTSHIFT Inverse FFT shift.
inverter.m: %% Inverses of Matrices
etc.
```

From this list one can see that the required function is `inv`. Syntax may then be got from `help inv`.

1.2.8 The Path Browser

MatLab comes with a large number of M-files in various directories.

1. When MatLab encounters a name it looks first to see if it is a variable name.
2. It then searches for the name as an M-file in the current directory. (This is one of the reasons to ensure that the program starts in the current directory.
3. It then searches for an M-file in the directories in the search path.

If one of your variables has the same name as an M-file or a MatLab instruction you will not be able to access that M-file or MatLab instruction. This is a common cause of problems.

The MatLab search path can be added to or changed at any stage by selecting `Desktop Tools|Path` from the Start Button. Path related functions include

`addpath` Adds a directory to the MatLab search path

`path` Display MatLab search path

`parh2rc` Adds the current path to the MatLab search path

`rmpath` Remove directory from MatLab search path

The command `cd` changes the current working directory

1.2.9 Miscellaneous Commands

Note the following MatLab commands

`clc` Clears the contents of the Command Window

`clf` - Clears the contents of the Figure Window

If MATLAB appears to be caught in a loop and is taking too long to finish a command it may be aborted by `^C` (Hold down the Ctrl key and press C). MATLAB will then return to the command prompt

`diary filename` After this command all input and most output is echoed to the specified file. The commands `diary off` and `diary on` will suspend and resume input to the diary (log) file.

2 Vectors, Matrices and Arrays

The basic variable in MatLab is an Array. (The numbers entered earlier can be regarded as (1×1) arrays, Column vectors as $(n \times 1)$ arrays and matrices as $(n \times m)$ arrays. MATLAB can also work with multidimensional arrays.

2.1 A Sample MATLAB session

It is recommended that you work through the following sitting at a PC with MATLAB running and enter the commands in the Command window. Most of the calculations involved are simple and they can be checked with a little mental arithmetic.

2.1.1 Entering Matrices

```
>> x=[1 2 3 4] % assigning values to a (1 by 4) matrix (row vector)
x =
        1        2        3        4
>> x=[1; 2; 3; 0]  % A (4 by 1) (row) vector
x =
        1
        2
        3
        4
>> x=[1,2,3;4,5,6] % (2 by 3) matrix
x =
        1        2        3
        4        5        6
>> x=[]  %Empty array
x = []
%*****************************
```

2.1.2 Basic Matrix operations

. The following examples are simple. Check the various operations and make sure that you understand them. This will also help you revise some matrix algebra which you will need for your theory.

```
>> x=[1 2;3 4]
x =
        1        2
        3        4

>> y=[3 7;5 4]
y =
        3        7
        5        4

>> x+y  %addition of two matrices - same dimensions
ans =
        4        9
        8        8

>> y-x  %matrix subtraction
ans =
        2        5
        2        0

>> x*y  % matrix multiplication
ans =
       13       15
       29       37
```

Note that when matrices are multiplied their dimensions must conform. The number of columns in the first matrix must equal the number of rows in the second otherwise MatLab returns an error. Try the following example. When adding matrices a similar error will be reported if the dimensions do not match

```
>> x=[1 2;3 4]
x =
     1     2
     3     4
>> z=[1,2]
z =
     1     2
>> x*z
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

```
>> inv(x) % find inverse of a matrix
ans =

  -2.00000   1.00000
   1.50000  -0.50000

>> x*inv(x) % verify inverse
ans =

  1.00000  0.00000
  0.00000  1.00000

>> y*inv(x) % multiply y by the inverse of x
ans =

   4.50000  -0.50000
  -4.00000   3.00000

>> y/x % alternative expression
ans =

   4.50000  -0.50000
  -4.00000   3.00000

>> inv(x)*y pre-multiply y by the inverse of x
ans =
  1.0e+01  *

  -0.10000  -1.00000
   0.20000   0.85000

>> x\y  % alternative expression - different algorithm - better for regression
ans =

  1.0e+01  *

  -0.10000  -1.00000
   0.20000   0.85000
```

2.1.3 Kronecker Product

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1m}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \dots & a_{2m}\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ a_{n1}\boldsymbol{B} & a_{n2}\boldsymbol{B} & \dots & a_{nm}\boldsymbol{B} \end{pmatrix}$$

```
x>> x=[1 2;3 4]

x =
     1     2
     3     4

>> I=eye(2,2)
I =
     1     0
     0     1
>> kron(x,I)
ans =

     1     0     2     0
     0     1     0     2
     3     0     4     0
     0     3     0     4

>> kron(I,x)

ans =
     1     2     0     0
     3     4     0     0
     0     0     1     2
     0     0     3     4
```

2.1.4 Examples of number formats

```
>> x=12.345678901234567;
>> format loose %includes blank lines to space output
>> x

x =

   12.3457

>> format compact %Suppress blank lines
>> x
x =
   12.3457
>> format long %14 digits after decimal
>> x
x =
  12.34567890123457
>> format short e % exponential or scientific format
>> x
x =
  1.2346e+001
>> format long e
>> x
x =
    1.234567890123457e+001
>> format short g % decimal or exponential
>> x
x =
       12.346
>> format long g
>> x
x =
          12.3456789012346
>> format bank % currency format (2 decimals)
>> x
```

```
x =
         12.35
```

2.1.5 fprintf function

```
>> fprintf('%6.2f\n', x )
 12.35
>> fprintf('%6.3f\n', x )
12.346
>> fprintf('The number is %6.4f\n', x )
The number is 12.3457
```

Here `fprintf` prints to the command window according to the format specification '%6.4f\n'. In this format specification the % indicates the start of a format specification. There will be at least 6 digits displayed of which 4 will be decimals in floating point (f). The \n indicates that the curser will then move to the next line. For more details see page 41.

2.1.6 element by element operations

```
% .operations
>> x=[1 2;3 4];
>> y=[3 7;5 4]
>> x .* y  %element by element multiplication
ans =

        3       14
       15       16

>> y ./ x  %element by element division
ans =

  3.00000  3.50000
  1.66667  1.00000

>> z=[3 7;0 4];
>> x./z
Warning: Divide by zero.
ans =
    0.3333    0.2857
       Inf    1.0000

%mixed scalar matrix operations
```

```
>> a=2;
>> x+a
ans =
        3        4
        5        6

>> x-a
ans =
        -1         0
         1         2

>> x*2
ans =
        2        4
        6        8

>> x/2
ans =
  0.50000  1.00000
  1.50000  2.00000

%exponents
% x^a is x^2 or x*x i.e.

>> x^a
ans =
        7       10
       15       22h

% element by element exponent

>> z = [1 2;2 1]
>> x .^ z
ans =
        1        4
        9        4
```

2.1.7 miscellaneous functions

Some functions. Operate element by element

```
>> exp(x)
```

```
ans =
  1.0e+01  *
  0.27183  0.73891
  2.00855  5.45982

>> log(x)
ans =
  0.00000  0.69315
  1.09861  1.38629



>> sqrt(x)
ans =
  1.00000  1.41421
  1.73205  2.00000
```

Using negative numbers in the argument of logs and square-roots produces an error in many other packages. MATLAB returns complex numbers. Take care!! This is mathematically correct but may not be what you want.

```
>> z=[1 -2]
z =
     1    -2
>> log(z)
ans =
        0             0.6931 + 3.1416i

>> sqrt(z)
ans =
   1.0000                  0 + 1.4142i

>> x-[1 2;3 4]
ans =
     0     0
     0     0
>> % determinant
>> det(x)
ans =
    -2

>> %trace
>> trace(x)
ans =
     5
```

The function `diag(`$\boldsymbol{X}$`)` where $\boldsymbol{X}$ is a square matrix puts the diagonal of $\boldsymbol{X}$ in a matrix. The function `diag(`$\boldsymbol{Z}$`)` where $\boldsymbol{Z}$ is a column vector outputs a matrix with diagonal $\boldsymbol{Z}$ and zeros elsewhere

```
>> z=diag(x)
z =
     1
     4



>> u=diag(z)
u =
     1     0
     0     4

% Rank of a matrix
>> a=[2 4 6 9
3 2 5 4
2 1 7 8]
a =
     2     4     6     9
     3     2     5     4
     2     1     7     8

>> rank(a)
ans =
     3
```

`sum(`$\boldsymbol{A}$`)` returns sums along different dimensions of an array. If $\boldsymbol{A}$ is a vector, `sum`($\boldsymbol{A}$) returns the sum of the elements. If $\boldsymbol{A}$ is a matrix, `sum`($\boldsymbol{A}$) treats the columns of $\boldsymbol{A}$ as vectors, returning a row vector of the sums of each column.

```
>> x=[1 2 3 4]


x =
     1     2     3     4
>> sum(x)
ans =
    10
>> sum(x')
ans =
    10
>> x=[1 2;3 4]
x =
     1     2
     3     4
>> sum(x)
ans =
     4     6
```

The function `reshape(`$\boldsymbol{A}$`,m,n)` returns the $m \times n$ matrix $\boldsymbol{B}$ whose elements are taken column-wise from $\boldsymbol{A}$. An error results if $\boldsymbol{A}$ does not have exactly mn elements

```
>> x=[1 2 3 ; 4 5 6]
x =
     1     2     3
     4     5     6
>> reshape(x,3,2)
ans =
     1     5
     4     3
     2     6
```

`blkdiag(`$\boldsymbol{A}$`,`$\boldsymbol{B}$`,`$\boldsymbol{C}$`,)` constructs a block diagonal matrix from the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ $\boldsymbol{c}$ etc.

```
a =
     1     2
     3     4
>> b=5
b =
     5
>> c=[6 7 8;9 10 11;12 13 14]
c =
     6     7     8
     9    10    11
    12    13    14
>> blkdiag(a,b,c)
ans =
     1     2     0     0     0     0
     3     4     0     0     0     0
     0     0     5     0     0     0
     0     0     0     6     7     8
     0     0     0     9    10    11
     0     0     0    12    13    14
```

This is only a small sample of the available functions

eigenvalues and eigenvectors

```
>> A=[54    45    68
    45    50    67
    68    67    95]
A =
    54    45    68
    45    50    67
    68    67    95


>> eig(A)
ans =
    0.4109
    7.1045
  191.4846

>> [V,D]=eig(A)
V =
    0.3970    0.7631    0.5100
    0.5898   -0.6378    0.4953
   -0.7032   -0.1042    0.7033
D =
    0.4109         0         0
         0    7.1045         0
         0         0  191.4846

>> Test=A*V
Test =
    0.1631    5.4214   97.6503
    0.2424   -4.5315   94.8336
   -0.2890   -0.7401  134.6750

>> Test ./ V
ans =
    0.4109    7.1045  191.4846
    0.4109    7.1045  191.4846
    0.4109    7.1045  191.4846
>>
```

2.1.8 sequences

colon operator (:) `first:increment:last` is a sequence with first element first second element first+ increment and continuing while entry is less than last.

```
>> [1:2:9]
ans =
        1        3        5        7        9

>> [2:2:9]
ans =
        2        4        6        8

>> [1:4]
ans =
        1        2        3        4


>> [1:4]'
ans =

        1
        2
        3
        4

%Transpose of a vector

>> x
x =
        1        2
        3        4

>> x'
ans =
        1        3
        2        4
```

2.1.9 Creating Special Matrices

```
% creating an Identity Matrix and matrices of ones and zeros

>> x=eye(4)
```

```
x =
        1        0        0        0
        0        1        0        0
        0        0        1        0
        0        0        0        1

>> x=ones(4)
x =
        1        1        1        1
        1        1        1        1
        1        1        1        1
        1        1        1        1

>> x=ones(4,2)
x =
        1        1
        1        1
        1        1
        1        1


>> x=zeros(3)
x =
        0        0        0
        0        0        0
        0        0        0

>> x=zeros(2,3)
x =
        0        0        0
        0        0        0

>> size(x)
ans =
        2        3
```

2.1.10 Random number generators

There are two random number generators in standard MATLAB.

`rand` generates uniform random numbers on [0,1)

`randn` generates random numbers from a normal distribution with zero mean and unit variance.

```
>> x=rand(5)
x =
  0.81551  0.55386  0.78573  0.05959  0.61341
  0.58470  0.92263  0.78381  0.80441  0.20930
  0.70495  0.89406  0.11670  0.45933  0.05613
  0.17658  0.44634  0.64003  0.07634  0.14224
  0.98926  0.90159  0.52867  0.93413  0.74421

>> x=rand(5,1)
x =
  0.21558
  0.62703
  0.04805
  0.20085
  0.67641

>> x=randn(1,5)
x =
   1.29029   1.82176  -0.00236   0.50538  -1.41244
```

2.1.11 Extracting parts of a matrix, Joining matrices together to get a new larger matrix

```
>> arr1=[2 4 6 8 10];
>> arr1(3)
ans = 6

>> arr2=[1, 2, -3;4, 5, 6;7, 8, 9]
arr2 =
         1         2        -3
         4         5         6
         7         8         9

>> arr2(2,2)
ans = 5

>> arr2(2,:)
ans =
        4        5        6

>> arr2(2,1:2)
```

```
ans =
        4         5

>> arr2(end,2:end)
ans =
        8         9
```

2.1.12 Using sub-matrices on left hand side of assignment

```
>> arr4=[1 2 3 4;5 6 7  8 ;9 10 11 12]
arr4 =
        1         2         3         4
        5         6         7         8
        9        10        11        12

>> arr4(1:2,[1,4])=[20,21;22 23]
arr4 =
       20         2         3        21
       22         6         7        23
        9        10        11        12



>> arr4=[20,21;22 23]
arr4 =
       20        21
       22        23

>> arr4(1:2,1:2)=1
arr4 =
     1     1
     1     1
>> arr4=[1 2 3 4;5 6 7  8 ;9 10 11 12]
arr4 =
        1         2         3         4
        5         6         7         8
        9        10        11        12

>> arr4(1:2,1:2)=1
arr4 =
        1         1         3         4
        1         1         7         8
        9        10        11        12
```

2.1.13 Stacking Matrices

```
>> x=[1 2;3 4]
x =
        1        2
        3        4

>> y=[5 6; 7 8]
y =
        5        6
        7        8

>> z=[x,y,(15:16)'] % join matrices side by side
z =

        1        2        5        6       15
        3        4        7        8       16

>> z=[x',y',(15:16)']' % Stack matrices vertically
z =
        1        2
        3        4
        5        6
        7        8
       15       16
```

See also the help files for the MatLab commands cat, `horzcat` and `vertcat`

2.1.14 Special Values

```
>> pi
pi = 3.1416
>> exp(1) %
e = 2.7183
>> clock
ans =
  1.0e+03  *
  2.00500  0.00100  0.01300  0.02200  0.01100  0.01932
  YEAR     Month    Day      hours    Minutes  Seconds
```

2.2 Examples

Work through the following examples using MATLAB.

1. let $\boldsymbol{A} = \begin{pmatrix} 3 & 0 \\ 5 & 2 \end{pmatrix}$ and $\boldsymbol{B} = \begin{pmatrix} 1 & 4 \\ 4 & 7 \end{pmatrix}$ Use MATLAB to calculate.

   (a) $\boldsymbol{A} + \boldsymbol{B}$

   (b) $\boldsymbol{A} - \boldsymbol{B}$

   (c) $\boldsymbol{AB}$

   (d) $\boldsymbol{AB}^{-1}$

   (e) $\boldsymbol{A}/\boldsymbol{B}$

   (f) $\boldsymbol{A}\backslash\boldsymbol{B}$

   (g) $\boldsymbol{A}.*\boldsymbol{B}$

   (h) $\boldsymbol{A}./\boldsymbol{B}$

   (i) $\boldsymbol{A} \otimes \boldsymbol{B}$

   (j) $\boldsymbol{B} \otimes \boldsymbol{A}$

   Use pen, paper and arithmetic to verify that your results are correct.

2. Let $\boldsymbol{A} = \begin{pmatrix} 1 & 4 & 3 & 7 \\ 2 & 6 & 8 & 3 \\ 1 & 3 & 4 & 5 \\ 4 & 13 & 15 & 15 \end{pmatrix}$ Use the MatLab function to show that the rank of A is three. Why is it not four?

3. Solve $\boldsymbol{Ax} = \boldsymbol{a}$ for $\boldsymbol{x}$ where $\boldsymbol{A} = \begin{pmatrix} 1 & 4 & 3 & 7 \\ 2 & 6 & 8 & 3 \\ 1 & 3 & 4 & 5 \\ 2 & 1 & 7 & 6 \end{pmatrix}$ and $\boldsymbol{a} = \begin{pmatrix} 14 \\ 8 \\ 10 \\ 18 \end{pmatrix}$

4. Generate $\boldsymbol{A}$ which is a 4×4 matrix of uniform random numbers. Calculate the trace and determinant of A. Use MATLAB to verify that

   (a) The product of the eigenvalues of $\boldsymbol{A}$ is equal to the determinant of $\boldsymbol{A}$

   (b) The sum of the eigenvalues of $\boldsymbol{A}$ is equal to the trace of $\boldsymbol{A}$. (You might find the MATLAB functions `sum()` and **prod()** helpful - please see the relevant help files). Do these results hold for an arbitrary matrix A.

5. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two 4×4 matrices of independent N(0,1) random numbers. If tr($\boldsymbol{A}$) is the trace of $\boldsymbol{A}$. Show that

   (a) $\text{tr}(\boldsymbol{A} + \boldsymbol{B}) = \text{tr}(\boldsymbol{A}) + \text{tr}(\boldsymbol{B})$

   (b) $\text{tr}(4\boldsymbol{A}) = 4\text{tr}(\boldsymbol{A})$

   (c) $\text{tr}(\boldsymbol{A'}) = \text{tr}(\boldsymbol{A})$

   (d) $\text{tr}(\boldsymbol{BA}) = \text{tr}(\boldsymbol{AB})$

   Which of these results hold for arbitrary matrices? Under what conditions would they hold for non-square matrices?

2.3 Regression Example

In this Example I shall use the instructions you have already learned to simulate a set of observations from a linear equation and use the simulated observations to estimate the coefficients in the equation. In the equation y_t is related to x_{2t} and x_{3t} according to the following linear relationship.

$$y_t = \beta_1 + \beta_2 x_{2t} + \beta_3 x_{3t} + \varepsilon_t, \quad t = 1, 2, \ldots, N$$

or in matrix notation

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where

- x_2 is a trend variable which takes the values (1,2, $\ldots$30)
- x_3 is a random variable with uniform distribution on $[3, 5]$
- ε_t are independent identically distributed normal random variables with zero mean and constant variance σ^2.
- $\beta_1 = 5$, $\beta_2 = 1$ and $\beta_3 = 0.1$ and ε_t are iidn(0,.04) ($\sigma^2 = 0.04$)

1. Verify that the model may be estimated by OLS.
2. Use MatLab to simulate 50 observations of each of x_3 and ε_t and thus of x_t.
3. Using the simulated values find OLS estimates of $\boldsymbol{\beta}$
4. Estimate the covariance matrix of $\boldsymbol{\beta}$ and thus the t-statistics for a test that the coefficients of $\boldsymbol{\beta}$ are zero.
5. Estimate the standard error or the estimate of $\boldsymbol{y}$
6. Calculate the F-statistic for the significance of the regression
7. Export the data to STATA and verify your result.
8. In a simulation exercise such as this two different runs will not produce the same result. Any answers submitted should be concise and short and should contain
9. A copy of the m-file used in the analysis. This should contain comments to explain what is being done
10. A short document giving the results of one simulation and any comments on the results. You might also include the regression table from the STATA analysis. This document should be less than one page in length.

A sample answer follows. First the program, then the output and finally some explanatory notes

```
% example1.m
% Regression Example Using Simulated Data
%John C Frain
%19 November 2006
%values for simulation
nsimul=50;
beta=[5,1,.1]';
%
% Step 1 Prepare and process data for X and y matrices/vectors
%
x1=ones(nsimul,1); %constant
x2=[1:nsimul]';    %trend
x3=rand(nsimul,1)*2 +3;  % Uniform(3,5)
X=[x1,x2,x3];
e=randn(nsimul,1)*.2;  % N(0,.04)
y= X * beta +e ;         %5*x1 + x2 + .1*x3 + e;
%
[nobs,nvar]=size(X);
%
% Estimate Model
```

Note that I have named my estimated variables `ols.betahat`, `ols.yhat`, `ols.resid` etc. The use of the `ols.` in front of the variable name has two uses. First if I want to do two different estimate I will call the estimates `ols1.` and `ols2.` or `IV.` etc. and I can easily put the in a summary table. Secondly this structure has a meaning that is useful in a more advanced use of MATLAB.

```
ols.betahat=(X'*X)\X'*y  % Coefficients
ols.yhat = X * ols.betahat;   % beta(1)*x1+beta(2)*x2+beta(3)*x;
ols.resid = y - ols.yhat; % residuals
ols.ssr = ols.resid'*ols.resid; % Sum of Squared Residuals
ols.sigmasq = ols.ssr/(nobs-nvar); % Estimate of variance
ols.covbeta=ols.sigmasq*inv(X'*X);  % Covariance of beta
ols.sdbeta=sqrt(diag(ols.covbeta));% st. dev of beta
ols.tbeta = ols.betahat ./ ols.sdbeta; % t-statistics of beta
ym = y - mean(y);
ols.rsqr1 = ols.ssr;
ols.rsqr2 = ym'*ym;
ols.rsqr = 1.0 - ols.rsqr1/ols.rsqr2; % r-squared
ols.rsqr1 = ols.rsqr1/(nobs-nvar);
ols.rsqr2 = ols.rsqr2/(nobs-1.0);
if ols.rsqr2 ~= 0;
ols.rbar = 1 - (ols.rsqr1/ols.rsqr2); % rbar-squared
else
ols.rbar = ols.rsqr;
end;
ols.ediff = ols.resid(2:nobs) - ols.resid(1:nobs-1);
ols.dw = (ols.ediff'*ols.ediff)/ols.ssr; % durbin-watson
fprintf('R-squared      = %9.4f \n',ols.rsqr);
fprintf('Rbar-squared   = %9.4f \n',ols.rbar);
fprintf('sigma^2        = %9.4f \n',ols.sigmasq);
fprintf('S.E of estimate= %9.4f \n',sqrt(ols.sigmasq));
fprintf('Durbin-Watson  = %9.4f \n',ols.dw);
fprintf('Nobs, Nvars    = %6d,%6d \n',nobs,nvar);
fprintf('*************************************************\n \n');
% now print coefficient estimates, SE, t-statistics and probabilities
%tout = tdis_prb(tbeta,nobs-nvar); % find t-stat probabilities - no
%tdis_prb in basic MATLAB - requires leSage toolbox
%tmp = [beta sdbeta tbeta tout];  % matrix to be printed
tmp = [ols.betahat ols.sdbeta ols.tbeta];  % matrix to be printed
% column labels for printing results
namestr = ' Variable';
bstring = '    Coef.';
sdstring= 'Std. Err.';
tstring = '  t-stat.';
cnames = strvcat(namestr,bstring,sdstring, tstring);
vname = ['Constant','Trend' 'Variable2'];
```

The `fprintf` is used to produce formatted output. See subsection 3.6

```
fprintf('%12s %12s %12s %12s \n',namestr, ...
    bstring,sdstring,tstring)
fprintf('%12s %12.6f %12.6f %12.6f \n',...
    '     Const',...
    ols.betahat(1),ols.sdbeta(1),ols.tbeta(1))
fprintf('%12s %12.6f %12.6f %12.6f \n',...
    '     Trend',...
    ols.betahat(2),ols.sdbeta(2),ols.tbeta(2))
fprintf('%12s %12.6f %12.6f %12.6f \n',...
    '      Var2',...
    ols.betahat(3),ols.sdbeta(3),ols.tbeta(3))
```

The output of this program should look like

```
R-squared      =    0.9998
Rbar-squared   =    0.9998
sigma^2        =    0.0404
S.E of estimate=    0.2010
Durbin-Watson  =    1.4445
Nobs, Nvars    =     50,     3
****************************************************

    Variable        Coef.    Std. Err.      t-stat.
       Const     4.804620     0.229091    20.972540
       Trend     0.996838     0.002070   481.655756
        Var2     0.147958     0.052228     2.832955
>>
```

Your answers will of course be different

Explanatory Notes

Most of your MATLAB scripts or programs will consist of three parts

1. **Get and Process data** Read in your data and prepare vectors or matrices of your left hand side ($\mathbf{y}$), Right hand side ($\mathbf{X}$) and Instrumental Variables ($\mathbf{Z}$)
2. **Estimation** Some form of calculation(s) like $\hat{\boldsymbol{\beta}} = (\boldsymbol{X'X}^{-1})\boldsymbol{X'y}$ implemented by a MATLAB instruction like

   ```
   betahat = (X'*X)\X*y
   ```

   (where `X` and `y` have been set up in the previous step) and estimate of required variances, covariances, standard errors etc.
3. **Report** Output tables and Graphs in a form suitable for inclusion in a report.
4. Run the program with a smaller number of replications (say 25) and see how the t-statistic on y_3 falls. Rerun it with a larger number of replications and see how it rises. Experiment to find how many observations are required to get a significant coefficient for y_3 often. Suggest a use of this kind of analysis.

2.4 Simulation – Sample Size and OLS Estimates

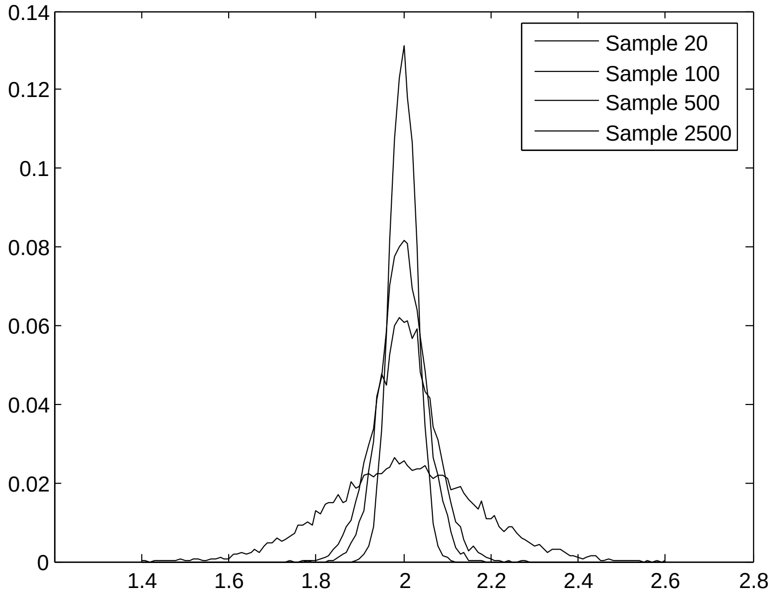
This exercise is a study of the effect of sample size on the estimates of the coefficient in an OLS regression. The x values for the regression have been generated as uniform random numbers on the interval [0,100). The residuals are simulated standardised normal random variables. The process is repeated for sample sizes of 20, 100 500 and 2500 simulation is repeated 10,000 times.

```
% example2.m
% MATLAB Simulation Example
%John C Frain
%19 November 2006
%
${
The data files x20.csv, x100.csv, x500.csv and x2500.csv
were generated  using the code below
$}
%Generate Data
x20 = 100*rand(20,1)
save('x20.csv','x20','-ASCII','-double')
x100 = 100*rand(100,1)
save('x100.csv','x100','-ASCII','-double')
x500 = 100*rand(500,1)
save('x500.csv','x500','-ASCII','-double')
x2500 = 100*rand(200,1)
save('x2500.csv','x2500','-ASCII','-double')
%}
clear
nsimul=10000;
BETA20=zeros(nsimul,1); % vector - results of simulations with 20 obs.
x=load('-ascii', 'x20.csv'); % load xdata
X=[ones(size(x,1),1),x]; % X matrix note upper case X
beta = [ 10;2];  % true values of coefficients
%
for ii = 1 : nsimul;
    eps = 20.0*randn(size(X,1),1); % simulated error term
    y = X * beta + eps; % y values
    betahat = (X'*X)\X'*y; % estimate of beta
    BETA20(ii,1)=betahat(2);
end
fprintf('Mean and st. dev of 20 obs simulation %6.3f %6.3f\n' ...
    ,mean(BETA20),std(BETA20))
%hist(BETA,100)

BETA100=zeros(nsimul,1);
x=load('-ascii', 'x100.csv'); % load xdata
X=[ones(size(x,1),1),x]; % X matrix note upper case X
beta = [ 10;2];  % true values of coefficients
%
for ii = 1 : nsimul;
```

```
    eps = 20.0*randn(size(X,1),1); % simulated error term
    y = X * beta + eps; % y values
    betahat = inv(X'*X)*X'*y; % estimate of beta
    BETA100(ii,1)=betahat(2);
end
fprintf('Mean and st. dev of 100 obs simulation %6.3f %6.3f\n', ...
    mean(BETA100),std(BETA100))

BETA500=zeros(nsimul,1);
x=load('-ascii', 'x500.csv'); % load xdata
X=[ones(size(x,1),1),x]; % X matrix note upper case X
beta = [ 10;2];  % true values of coefficients
%
for ii = 1 : nsimul;
    eps = 20.0*randn(size(X,1),1); % simulated error term
    y = X * beta + eps; % y values
    betahat = inv(X'*X)*X'*y; % estimate of beta
    BETA500(ii,1)=betahat(2);
end
fprintf('Mean and st. dev of 500 obs simulation %6.3f %6.3f\n', ...
    mean(BETA500),std(BETA500))

BETA2500=zeros(nsimul,1);
x=load('-ascii', 'x2500.csv'); % load xdata note use of lower case x as vector
X=[ones(size(x,1),1),x]; % X matrix note upper case X
beta = [ 10;2];  % true values of coefficients
%
for ii = 1 : nsimul;
    eps = 20.0*randn(size(X,1),1); % simulated error term
    y = X * beta + eps; % y values
    betahat = inv(X'*X)*X'*y; % estimate of beta
    BETA2500(ii,1)=betahat(2);
end
fprintf('Mean and st. dev of 2500 obs simulation %6.3f %6.3f\n', ...
    mean(BETA2500),std(BETA2500))

n=hist([BETA20,BETA100,BETA500,BETA2500],1.4:0.01:2.6);
plot((1.4:0.01:2.6)',n/nsimul);
h = legend('Sample 20','Sample 100','Sample 500','Sample 2500');
```

The output of this program will look like this. On your screen the graph will display coloured lines.

```
Mean and st. dev of 20 obs simulation  2.000  0.165
Mean and st. dev of 100 obs simulation  2.000  0.065
Mean and st. dev of 500 obs simulation  2.000  0.030
Mean and st. dev of 2500 obs simulation  1.999  0.049
```



2.5 Example – Macroeconomic Simulation with Matlab

Problem

This example is based on the macroeconomic system in Example 10.3 of Shone (2002). There are 10 equations in this economic model. The equations of the system are as follows

$$\begin{aligned}
c_t &= 110 + 0.75yd_t \\
yd_t &= y_t - tax_t \\
tax_t &= -80 + 0.2y_t \\
i_t &= -4r_t \\
g_t &= 330 \\
e_t &= c_t + i_t + g_t \\
y_t &= e_{t-1} \\
md_t &= 20 + 0.25y_t - 10r_t \\
ms_t &= 470 \\
md_t &= ms_t
\end{aligned}$$

While the aim in Shone (2002) is to examine the system algebraically, here we examine it numerically. Often this may be the only way to solve the system and Matlab is a suitable tool for this work. The model is too simple to be of any particular use in macroeconomics but it does allow one to illustrate the facilities offered by Matlab for this kind of work.

Initialise and Describe Variables

```
N = 15 ; % Number of periods for simulation
c = NaN * zeros(N,1); %real consumption
tax = NaN * zeros(N,1); %real tax
yd = NaN * zeros(N,1); %real disposible income
i = NaN * zeros(N,1); % real investment
g = NaN * zeros(N,1); % real government expenditure
e = NaN * zeros(N,1); % real expenditure
y = NaN * zeros(N,1); % real income
md = NaN * zeros(N,1); % real money demand
ms = NaN * zeros(N,1); %real money supply
r = NaN * zeros(N,1); % interest rate
```

Simulate

g and ms are the policy variables.

```
t=(1:N)'; % time variable
g = 330 * ones(N,1);
ms = 470 * ones(N,1);
y(1) = 2000;
```

The next step is to simulate the model over the required period. In this case this is achieved by a simple reordering of the equations and inverting the money demand equation to give an interest rate equation. In the general case we might need a routine to solve the set of non linear equations or some routine to maximise a utility function. Note that the loop stops one short of the full period and then does the calculations for the final period (excluding the income calculation for the period beyond the end of the sample under consideration).

```
for ii = 1:(N-1)
    tax(ii) = -80 + 0.2 * y(ii);
    yd(ii) = y(ii) - tax(ii);
    c(ii)  = 110 + 0.75 * yd(ii);
    md(ii) = ms(ii);
    r(ii) = (20 + 0.25* y(ii) -md(ii))/10; % inverting money demand
    i(ii) = 320 -4 * r(ii);
    e(ii) = c(ii) + i(ii) + g(ii);
    y(ii+1) = e(ii);
end

tax(N) = -80 + 0.2 * y(N);
yd(N) = y(N) - tax(N);
c(N)  = 110 + 0.75 * yd(N);
md(N) = ms(N);
r(N) = (20 +  0.25* y(N) -md(N))/10;
i(N) = 320 -4 * r(N);
e(N) = c(N) + i(N) + g(N);
```

Now output results and save y for later use. note that the system is in equilibrium. Note that in printing we use the transpose of base

```
base = [t,y,yd,c,g-tax,i,r];
fprintf('      t      y     yd      c  g-tax     i       r\n')
fprintf('%7.0f%7.0f%7.0f%7.0f%7.0f%7.0f%7.0f\n',base')
ybase = y;
```

```
      t      y     yd      c  g-tax      i      r
      1   2000   1680   1370     10    300      5
      2   2000   1680   1370     10    300      5
      3   2000   1680   1370     10    300      5
      4   2000   1680   1370     10    300      5
      5   2000   1680   1370     10    300      5
      6   2000   1680   1370     10    300      5
      7   2000   1680   1370     10    300      5
      8   2000   1680   1370     10    300      5
      9   2000   1680   1370     10    300      5
     10   2000   1680   1370     10    300      5
     11   2000   1680   1370     10    300      5
     12   2000   1680   1370     10    300      5
     13   2000   1680   1370     10    300      5
     14   2000   1680   1370     10    300      5
     15   2000   1680   1370     10    300      5
```

Revised Simulation

We increase g to 350 and examine the passage to the new equilibrium. Basically we run the same program with a different starting value for g.

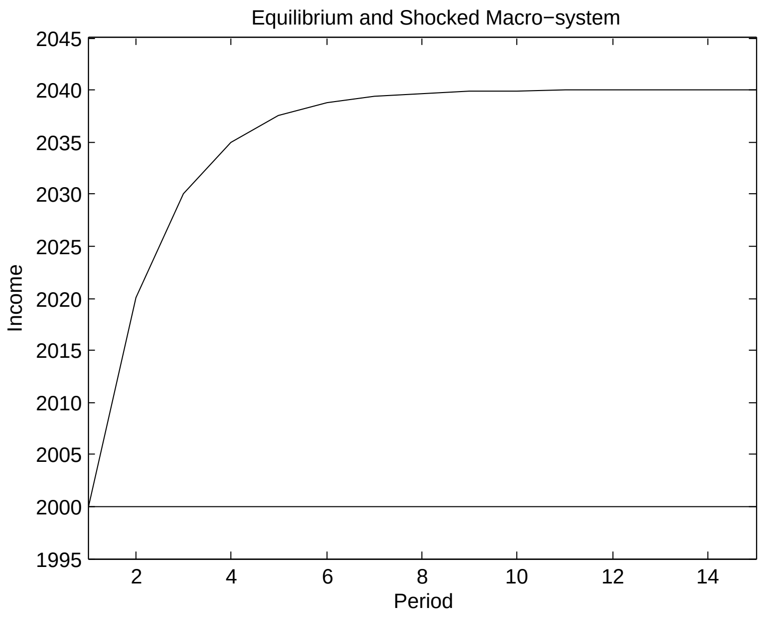
```
N = 15 ; % Number of periods for simulation
c = NaN * zeros(N,1); %real consumption
tax = NaN * zeros(N,1); %real tax
yd = NaN * zeros(N,1); %real disposible income
i = NaN * zeros(N,1); % real investment
g = NaN * zeros(N,1); % real government expenditure
e = NaN * zeros(N,1); % real expenditure
y = NaN * zeros(N,1); % real income
md = NaN * zeros(N,1); % real money demand
ms = NaN * zeros(N,1); %real money supply
r = NaN * zeros(N,1); % interest rate

% Policy Variables
g = 350 * ones(N,1);
ms = 470 * ones(N,1);
t=(1:N)';

y(1) = 2000;
for ii = 1:(N-1)
    tax(ii) = -80 + 0.2 * y(ii);
    yd(ii) = y(ii) - tax(ii);
    c(ii)  = 110 + 0.75 * yd(ii);
    md(ii) = ms(ii);
    r(ii) = (20 + 0.25* y(ii) -md(ii))/10; % inverting money demand
    i(ii) = 320 -4 * r(ii);
    e(ii) = c(ii) + i(ii) + g(ii);
    y(ii+1) = e(ii);
end

tax(N) = -80 + 0.2 * y(N);
yd(N) = y(N) - tax(N);
c(N)  = 110 + 0.75 * yd(N);
md(N) = ms(N);
r(N) = (20 +  0.25* y(N) -md(N))/10;
i(N) = 320 -4 * r(N);
e(N) = c(N) + i(N) + g(N);


policy = [t,y,yd,c,g-tax,i,r];
fprintf('      t      y     yd      c  g-tax     i       r\n')
fprintf('%7.0f%7.0f%7.0f%7.0f%7.0f%7.0f%7.0f\n',policy')
ypolicy =y;
```

```
     t      y     yd      c  g-tax      i        r
     1   2000   1680   1370     30    300        5
     2   2020   1696   1382     26    298        6
     3   2030   1704   1388     24    297        6
     4   2035   1708   1391     23    297        6
     5   2038   1710   1393     23    296        6
     6   2039   1711   1393     22    296        6
     7   2039   1712   1394     22    296        6
     8   2040   1712   1394     22    296        6
     9   2040   1712   1394     22    296        6
    10   2040   1712   1394     22    296        6
    11   2040   1712   1394     22    296        6
    12   2040   1712   1394     22    296        6
    13   2040   1712   1394     22    296        6
    14   2040   1712   1394     22    296        6
    15   2040   1712   1394     22    296        6
```

Now we compare results in a table and a graph. Note that income converges to a new limit.

```
fprintf('     t ybase ypolicy\n')
fprintf('%7.0f%7.0f%7.0f\n',[t,ybase, ypolicy]')

plot(t,[ybase,ypolicy])
title('Equilibrium and Shocked Macro-system')
xlabel('Period')
ylabel('Income')
axis([1 15 1995 2045])
```

```
     t  ybase ypolicy
     1   2000   2000
     2   2000   2020
     3   2000   2030
     4   2000   2035
     5   2000   2038
     6   2000   2039
     7   2000   2039
     8   2000   2040
     9   2000   2040
    10   2000   2040
    11   2000   2040
    12   2000   2040
    13   2000   2040
    14    2000    2040
    15    2000    2040
```



3 Data input/output

3.1 Native MatLab data files

The instruction `Save filename` saves the contents of the workspace in the file `'filename.mat'`. `save` used in the default manner saves the data in a binary format. The instruction `save filename, var1, var2` saves `var1` and `var2` in the file `filename.mat`. Similarly the commands `Load filename` and `load filename, var1, var2`. load the contents of `'filename.mat'` or the specified variables from the file into the workspace. In general .mat files are nor easily readable in most other packages. They are ideal for use within MATLAB and for exchange between MATLAB users. (note that there may be some incompatibilities between different versions of MATLAB). These .mat files are binary and can not be examined in a text editor.

`.mat` is the default extension for a MATLAB data file. If you use another extension, say `.ext` the option Save `mat filename.ext` should be used with the save and load commands. It is possible to use save and load to save and load text files but these instructions are very limited. If your data are in EXCEL or csv format the methods described below are better

3.2 Importing from Excel

The sample file `g10xrate.xls` contains daily observations on the exchange rates of G10 countries and we wish to analyse them with MATLAB. There are 6237 observations of each exchange rate in the columns of the EXCEL file. The easiest way to import these data into MATLAB is to use the `File|import data` wizard and follow the prompts. In this case the import wizard did not pick out the series names from the Excel file. I imported the entire data matrix as a matrix and extracted the individual series in MATLAB. One can save the data in MATLAB format for future use in MATLAB This is illustrated in the code below.

```
USXJPN = data(:,1);
USXFRA = data(:,2);
USXSUI = data(:,3);
USXNLD = data(:,4);
USXUK = data(:,5);
USXBEL = data(:,6);
USXGER = data(:,7);
USXSWE = data(:,8);
USXCAN = data(:,9);
USXITA = data(:,10);
save('g10xrate', 'USXJPN','USXFRA','USXSUI','USXNLD','USXUK','USXBEL','USXGER', ...
    'USXSWE','USXCAN','USXITA')
```

Note that I have listed the series to be saved as I did not wish to save the data matrix. The same effect could have been achieved with the `uiimport` command.

3.3 Reading from text files

The import wizard can also import many types of text file including the comma separated files we have used in `STATA`. The missing data code in Excel csv files is #NA. The version of MATLAB that i am using has problems reading this missing value code and it should be changed to NaN (the MATLAB missing value code) before importing csv data. In this case the import wizard recognised the column names. It is important that you check that all your data has been imported correctly.

The MATLAB functions `textscan` or `textread` can read various text files and allow a greater degree of flexibility than that available from `uiimport`. This flexibility is obtained at a cost of greater complexity. Details are given in the Help files. I would think that most users will not need this flexibility but it is there if needed.

3.4 Exporting data to EXCEL, STATA and other programs

The command `xlswrite('filename',M)` writes the matrix **M** to the file `filename` in the current working directory. If **M** is $n \times m$ the numeric values in the matrix are written to the first n row and m columns in the first sheet in the spreadsheet. The command `csvwrite('filename',M)` writes the matrix **M** to the file `filename` in the current working directory. You can use this file to transfer data to `STATA`. Alternatively export your Excel file from Excel in csv format.

3.5 Stat/Transfer

Another alternative is to use the Stat/Transfer package which allows the transfer of data files between a large number of statistical packages.

3.6 Formatted Output

The MATLAB function `fprintf()` may be used to produce formatted output on screen[1]. The following MATLAB program gives an example of the us of the `fprintf()` function.

Sample MATLAB program demonstrating Formatted Output

```
clear
degrees_c =10:10:100;
degrees_f = (degrees_c * 9 /5) + 32;
fprintf('\n\n Conversion from degrees Celsius \n');
fprintf('     to degrees Fahrenheit\n\n' );
fprintf('     Celsius  Fahrenheit\n');
for ii = 1:10;
    fprintf('%12.2f%12.2f\n',degrees_c(ii),degrees_f(ii));
end
%
fprintf(...
'\n\n%5.2f degrees Celsius is equivalent of %5.3f degrees fahrenheit\n', ...
    degrees_c(1),degrees_f(1))
```

Output of Sample MATLAB program demonstrating Formatted Output

```
 Conversion from degrees Celsius
```

[1]`fprintf()` is only one of a large number of C-style input/output functions in C. These allow considerable flexibility in sending formatted material to the screen of to a file. The MATLAB help files give details of the facilities available. If further information is required one might consult a standard test book on the C programming language

```
to degrees Fahrenheit

Celsius  Fahrenheit
  10.00       50.00
  20.00       68.00
  30.00       86.00
  40.00      104.00
  50.00      122.00
  60.00      140.00
  70.00      158.00
  80.00      176.00
  90.00      194.00
 100.00      212.00


10.00 degrees Celsius is equivalent of 50.000 degrees fahrenheit
```

Note the following

- The first argument of the `fprintf()` function is a kind of format statement included within ' marks.
- The remaining arguments are a list of variables or items to be printed separated by commas
- Within the format string there is text which is produced exactly as set down. There are also statements like `%m.nf` which produces a decimal number which is allowed m columns of output and has n places of decimals. These are applied in turn to the items in the list to be printed.
- This `f` format is used to output floating point numbers there are a considerable number or other specifiers to output characters, strings, and number in formats other than floating point.
- If the list to be printed is too long the formats are recycled.
- Not the use of \`n` which means skip to the next line. This is essential.

3.7 Producing material for inclusion in a paper

A considerable amount of the material in this note was produced from MATLAB m-files using the —**File**—**Publish to**— facilities in the MATLAB m-file editor which produces output in WORD, Powerpoint, LaTeX , HTML etc. for inclusion in papers, presentations etc. The facilities are described in the help files and may vary from version to version of Matlab.

To Access these facilities you must first turn them on in the Matlab Editor. This is done by —**Cell**—**Enable Cell Mode**— in the editor menu. Cell mode enables you to divide your m-file into cells or sections. (Do not confuse cell mode in the editor with cell data structures in Matlab. It is unfortunate that these two different concepts have the same name) Enabling cell mode adds a new set of buttons to the menu bar and enables a set of items in the cell menu item. The menu items allow one to

- Disable cell mode
- Evaluate current cell
- Evaluate current cell and advance
- Evaluate entire file
- Insert Cell Divider
- Insert Cell Dividers around Selection
- Insert Cell Markup
  - Cell Title
  - Descriptive Text
  - Bold Text
  - Monospaced Text
  - Preformatted Text
  - Bulleted Text
  - TEX Equation
- Next Cell
- Previous Cell

Section 2.5, including its graph, was subject to very minor editing before being added to this document. I have also used the facility to produce transparencies for lectures on MATLAB.

4 Decision and Loop Structures.

There are four basic control (Decision or Loop Structures) available in MATLAB

if statements The basic form of the `if` statement is

```
if conditions
    statements
end
```

The `statements` are only processed if the conditions are true The conditions can include the following operators

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

The `if` statement may be extended

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

in which case statements1 are used if conditions are true and statements2 if false. This `if` statement may be extended again

```
if conditions1
    statements1
elseif conditions2
    statements2
else
    statements3
end
```

with an obvious meaning (I hope).

for The basic form of the for group is

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

Here `expression` is probably a vector. `statements` is processed for each of the values in expression. The following example shows the use of a loop within a loop

```
>> for ii = 1:3
for jj=1:3
total=ii+jj;
fprintf('%d + %d = %d \n',ii,jj,total)
end
end
1 + 1 = 2
1 + 2 = 3
1 + 3 = 4
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
3 + 1 = 4
3 + 2 = 5
3 + 3 = 6
```

while The format of the `while` statement is

```
while conditions
    statements
end
```

The `while` statement has the same basic functionality as the `for` statement. The `for` statement will be used when one knows precisely when and how many times an operation will be repeated. The statements are repeated so long as conditions are true

switch An example of the use of the `switch` statement follows

```
switch p
    case 1
        x = 24
    case 2
        x = 19
    case 3
        x = 15
    otherwise
        error('p must be 1, 2 or 3')
end
```

Use matrix statements in preference to loops. Not only are they more efficient but they are generally easier to use. That said there are occasions where one can not use a matrix statement.

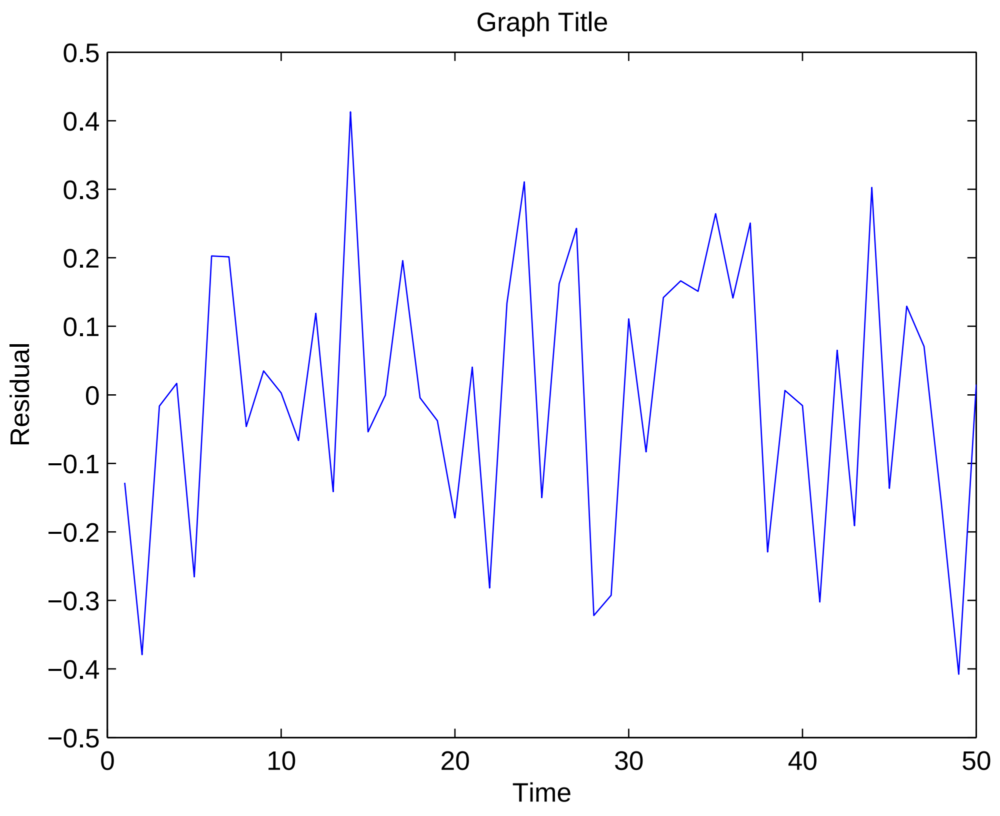
If you wish to fill the elements of a vector or matrix using a loop it is good practice to initialise the vector or matrix first. For example if you wish to fill a 100×20 matrix, $\boldsymbol{X}$, using a series of loops one could initialise the matrix using one of the following commands

```
X = ones(100,20)
X = zeros(100,20)
X = ones(100,20)*NaN
X = NaN(100,20)
```

5 Elementary Plots

Simple graphs can be produced easily in MatLab. The following sequence

```
%values for simulation
nsimul=50;
beta=[5,1,.1]';
%
x1=ones(nsimul,1); %constant
x2=[1:nsimul]';    %trend
x3=rand(nsimul,1)*2 +3;  % Uniform(3,5)
x=[x1,x2,x3];
e=randn(nsimul,1)*.2;  % N(0,.04)
y= x * beta +e ;         %5*x1 + x2 + .1*x3 + e;
%
[nobs,nvar]=size(x);
betahat=inv(x'*x)*x'*y %g
yhat = x * betahat;   % beta(1)*x1-beta(2)*x2-beta(3)*x;
resid = y - yhat;
plot(x2,resid)
title('Graph Title')
xlabel('Time')
ylabel('Residual')
```



repeats the earlier OLS simulation, opens a graph window, draws a graph of the residuals against the trend in the ols-simulation exercise, puts a title on the graph and labels the x and y axes. The vectors x2 and resid must have the same dimensions. This graph was saved in eps format and imported into this document.

6 Systems of Regression Equations

6.1 Using Matlab to estimate systems of regression equations

This section contains two examples of the estimation of systems of equations. The first is an examination of the classic Grunfeld investment data set. Many textbooks use this dataset to illustrate various features of system estimation. Green (2000) is the source of the data used here. Later editions of this book also examine these data but in less detail.

The MATLAB output also includes corresponding analysis using the le Sage econometrics package which is covered in section 8 of this note. As an exercise the user might extend the analysis to include various Likelihood Ratio tests of the restrictions imposed by the various estimation procedures.

Analysis of Grunfeld Investment data

Introduction

The basic system model that we are looking at here is

$$y_{ti} = \boldsymbol{X}_{ti}\boldsymbol{\beta}_i + \varepsilon_{ti}$$

where $1 \leq i \leq M$ represents the individual agent or country or item for which we are estimating some equation and $1 \leq t \leq T$ represents the t^{th} measurement on the i^{th} unit. We assume that the variance of ε_{ti}, σ_i^2 is constant for $1 \leq t \leq T$. Each $\boldsymbol{X}_i$ is $T \times k_i$. We may write these equations

$$\begin{aligned}
\boldsymbol{y}_1 &= \boldsymbol{X}_1\boldsymbol{\beta}_1 + \boldsymbol{\varepsilon_1} \\
\boldsymbol{y}_2 &= \boldsymbol{X}_2\boldsymbol{\beta}_2 + \boldsymbol{\varepsilon_2} \\
&\ldots \\
\boldsymbol{y}_M &= \boldsymbol{X}_M\boldsymbol{\beta}_M + \boldsymbol{\varepsilon_M}
\end{aligned}$$

In this section we will assume that $\boldsymbol{X}$ is exogenous. By imposing various cross-equation restrictions on the $\boldsymbol{\beta}_i$ and the covariances of the ε_{ti} we obtain a variety of estimators (e.g. Pooled OLS, Equation be Equation OLS, SUR).

The variables included in the Grunfeld analysis are

- FIRM : There are 10 firms
- YEAR : Data are from 1935 to 1954 (20 years)
- I : Gross Investment
- F : Value of Firm
- C : Stock of Plant and Equipment

For more details see Green (2000, 2008) or the original references listed there.

- To reduce the amount of detail we shall restrict analysis to 5 firms
- Firm no 1 : GM - General Motors
- Firm no 4 : GE - general electric
- Firm no 3 : CH - Chrysler
- Firm no 8 : WE - Westinghouse
- Firm no 2 : US - US Steel

To start the analysis is use the MATLAB Import data using |File|Import Data]. The test names on the data set are not imported as I wish to define these myself. This sets up a matrix containing data. I save data in Matlab form the first time. Later I use

```
load data; %
```

to reload the data as below

Load and Generate data

```
load data
Y_GM = data(1:20, 3); % I
X_GM = [ones(20,1),data(1:20,[4 5])]; % constant F C
Y_GE = data(61:80, 3); % I
X_GE = [ones(20,1),data(61:80,[4 5])]; % constant F C
Y_CH = data(41:60, 3); % I
X_CH = [ones(20,1),data(41:60,[4 5])]; % constant F C
Y_WE = data(141:160, 3); % I
X_WE = [ones(20,1),data(141:160,[4 5])]; % constant F C
Y_US = data(21:40, 3); % I
X_US = [ones(20,1),data(21:40,[4 5])]; % constant F C
```

We now estimate the coefficients imposing various restrictions. Each estimation involves the following steps

1. Set up the required $\boldsymbol{y}$ and $\boldsymbol{X}$ matrices.
2. Estimate the required coefficients.
3. Estimate standard errors, t-statistics etc.
4. Report.

Pooled OLS

The restrictions imposed by Pooled OLS are that corresponding coefficients are the same across equations. We also assume that the variance of the disturbances is constant across equations.[2] Thus, in this case $k_i = k$, for all i We can therefore assume that each observation on each unit is one more observation from the same single equation system. We may write the system as follows

$$\begin{array}{cccccc} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \cdots \\ \boldsymbol{y}_M \end{pmatrix} & = & \begin{pmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_2 \\ \cdots \\ \boldsymbol{X}_M \end{pmatrix} & \boldsymbol{\beta} & + & \begin{pmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \cdots \\ \boldsymbol{\varepsilon}_M \end{pmatrix} \\ (MT \times 1) & & (MT \times k) & (k \times 1) & & (MT \times 1) \end{array}$$

or, more compactly, using the obvious notation

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

and $\boldsymbol{\beta}$ may be estimated by $\hat{\boldsymbol{\beta}} = (\boldsymbol{X'}\boldsymbol{X})^{-1}\boldsymbol{X'}\boldsymbol{y}$ etc. This is implemented in MATLAB as follows –

[2]We could of course relax this condition and estimate Heteroskedastic Consistent Standard Errors

```
Y = [Y_GM', Y_GE', Y_CH', Y_WE', Y_US']'; % leave out ; for testing if
%                  necessary delete during run or output will be unreadable
X = [X_GM', X_GE', X_CH', X_WE', X_US']';

pols.beta = (X'*X)\X'*Y;
pols.uhat = Y - X*pols.beta ;
pols.sigsq = (pols.uhat'*pols.uhat)/(size(X,1)-size(X,2));%(T-k)
pols.sdbeta  = sqrt(diag(inv(X'*X))*pols.sigsq);
pols.tbeta = pols.beta ./ pols.sdbeta;
pols.se = sqrt(pols.sigsq);
label = ['Constant  '; 'F         '; 'C         '];
disp('OLS Results using stacked matrices')
disp('                coef        sd    t-stat')
for ii=1:size(X,2)
fprintf('%s%10.4f%10.4f%10.4f\n',label(ii,:),pols.beta(ii),pols.sdbeta(ii), pols.tbeta(ii))
end
fprintf('Estimated Standard Error %10.4f\n\n\n',pols.se)

OLS Results using stacked matrices
               coef        sd    t-stat
Constant    -47.0237   21.6292   -2.1741
F             0.1059    0.0114    9.2497
C             0.3014    0.0437    6.8915
Estimated Standard Error   128.1429
%
% Verification using Lesage package
%
pooled = ols(Y, X);
vnames= ['I         ';
         'Constant  ';
         'F         ';
         'C         '];
prt(pooled,vnames)


\begin{verbatim}
Ordinary Least-squares Estimates
Dependent Variable =       I
R-squared      =    0.7762
Rbar-squared   =    0.7716
sigma^2        = 16420.6075
Durbin-Watson  =    0.3533
```

```
Nobs, Nvars    =    100,     3
***************************************************************
Variable        Coefficient      t-statistic    t-probability
Constant         -47.023691        -2.174080         0.032132
F                  0.105885         9.249659         0.000000
C                  0.301385         6.891475         0.000000
```

Equation by equation OLS

This section assumes that the coefficients vary across units. (In panel data estimation we assume that only the constant terms vary across units). We also assume that there is no contemporaneous correlation between the disturbances in the system. We may write the system of equations as

$$\begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_M \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{X}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{X}_M \end{pmatrix} \boldsymbol{\beta} + \begin{pmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \vdots \\ \boldsymbol{\varepsilon}_M \end{pmatrix}$$

or more compactly using the obvious substitutions

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{y}$ and $\boldsymbol{\varepsilon}$ are $TM \times 1$, $\boldsymbol{X}$ is $TM \times kM$ and $\boldsymbol{\beta}$ is $kM \times 1$. $\boldsymbol{y}$, $\boldsymbol{\varepsilon}$, and $\boldsymbol{\beta}$ are stacked versions of $\boldsymbol{y}_i$, $\boldsymbol{\varepsilon}_i$, and $\boldsymbol{\beta}_i$. The variance of $\boldsymbol{\varepsilon}$ is given by

$$\begin{aligned} \Omega &= E\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}'\right] \\ &= \begin{pmatrix} \sigma_1^2 \boldsymbol{I}_T & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sigma_2^2 \boldsymbol{I}_T & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \sigma_M^2 \boldsymbol{I}_T \end{pmatrix} \\ &= \begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_M^2 \end{pmatrix} \otimes \boldsymbol{I}_T \end{aligned}$$

The coding of this example should be relatively clear. Perhaps the most difficult part is the estimation of the variances. The procedure here is very similar to that used in the first step of the SUR estimation procedure except that the contemporaneous correlation is used to improve the estimates. It should be noted that, in this case different variables are likely to be used in different equations, The only change required is in the calculation of the $\boldsymbol{X}$ matrix.

```
% Y as before
X=blkdiag(X_GM ,X_GE , X_CH , X_WE , X_US);
eqols.beta = (X'*X)\X'*Y;
eqols.uhat = Y - X*eqols.beta ;
eqols.temp = reshape(eqols.uhat,size(X_GM,1),5); %residuals for
                                               % each firm in a column
eqols.sigsq1 =eqols.temp'*eqols.temp/(size(X_GM,1)-size(X_GM,2));
eqols.sigsq = diag(diag(eqols.sigsq1)); % Remove non-diagonal elements
%eqols.sdbeta = sqrt(diag(inv(X'*X)*X'*kron(eye(size(X_GM,1)),eqols.sigsq)*X*inv(X'*X)));
eqols.covarbeta = inv(X'*X)*kron(eqols.sigsq,eye(3));
eqols.sdbeta = diag(sqrt(eqols.covarbeta));
eqols.tbeta=eqols.beta ./ eqols.sdbeta;
eqols.se=sqrt(diag(eqols.sigsq));
%
% Write results
%
disp('OLS equation by equation using stacked matrices')
disp('OLS estimates GE equation')

firm = ['GE';
        'GM';
        'CH';
        'wE';
        'US'];
for jj = 1:5 % Loop over firms
    fprintf('\n\n\n')
    disp('                coef        sd    t-stat')
    for ii=1:3 %Loop over coefficients
        fprintf('%10s%10.4f%10.4f%10.4f\n',label(ii), ...
            eqols.beta(ii+(jj-1)*3),eqols.sdbeta(ii+(jj-1)*3), ...
            eqols.tbeta(ii+(jj-1)*3))
    end
    fprintf('Standard Error is %10.4f\n',eqols.se(jj));
end
```

```
OLS equation by equation using stacked matrices
OLS estimates GE equation




               coef        sd    t-stat
         C -149.7825  105.8421   -1.4151
         F    0.1193    0.0258    4.6172
         C    0.3714    0.0371   10.0193
Standard Error is    91.7817




                coef         sd    t-stat
         C   -6.1900   13.5065   -0.4583
         F    0.0779    0.0200    3.9026
         C    0.3157    0.0288   10.9574
Standard Error is    13.2786




                coef         sd    t-stat
         C   -9.9563   31.3742   -0.3173
         F    0.0266    0.0156    1.7057
         C    0.1517    0.0257    5.9015
Standard Error is    27.8827




                coef         sd    t-stat
         C   -0.5094    8.0153   -0.0636
         F    0.0529    0.0157    3.3677
         C    0.0924    0.0561    1.6472
Standard Error is    10.2131




                coef         sd    t-stat
         C  -49.1983  148.0754   -0.3323
         F    0.1749    0.0742    2.3566
         C    0.3896    0.1424    2.7369
Standard Error is    96.4345
```

Verify using le Sage Toolbox

```
olsestim=ols(Y_US,X_US);
prt(olsestim, vnames);
```

```
Ordinary Least-squares Estimates
Dependent Variable =       I
R-squared      =    0.4709
```

```
Rbar-squared    =    0.4086
sigma^2         = 9299.6040
Durbin-Watson   =    0.9456
Nobs, Nvars     =     20,     3
***************************************************************
Variable        Coefficient      t-statistic    t-probability
Constant         -49.198322        -0.332252         0.743761
F                  0.174856         2.356612         0.030699
C                  0.389642         2.736886         0.014049
```

SUR Estimates

Suppose that we have a random sample of households and we are have time series data on expenditure on holidays (y_{it}) and relevant explanatory variables. Suppose that we have sufficient data to estimate a single equation for each person in the sample. We also assume that there is no autocorrelation in each equation (often a rather heroic assumption). During the peak of the business cycle it is likely that many of the persons in the sample spend above what they do at the trough. Thus it is likely that there will be contemporaneous correlation between the errors in the system.

$$E[\varepsilon_{ti}\varepsilon_{sj}] = \begin{cases} \sigma_{ij} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Thus we may write the contemporaneous covariance matrix ($\boldsymbol{\Sigma}$) as

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1M} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{M1} & \sigma_{M2} & \cdots & \sigma_{MM} \end{pmatrix}$$

The total covariance matrix is, in this case, given by

$$\begin{aligned} \Omega &= \begin{pmatrix} \boldsymbol{\Sigma} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Sigma} \end{pmatrix} \\ &= \boldsymbol{\Sigma} \otimes \boldsymbol{I}_T \end{aligned}$$

If $\boldsymbol{\Omega}$ were known we would use GLS to get optimum estimates of $\boldsymbol{\beta}$. In this case we can obtain a consistent estimate of $\boldsymbol{\Sigma}$ from the residuals in the equation by equation OLS estimate that we have just completed. We can then use this consistent estimate in Feasible GLS.

```
Omega = kron(eqols.sigsq1,eye(20,20)); % Page page 256
eqsur.beta= inv(X'*inv(Omega)*X)*X'*inv(Omega)*Y;
eqsur.yhat = X * eqsur.beta;
eqsur.uhat = Y - eqsur.yhat;
eqsur.temp = reshape(eqsur.uhat,20,5);
eqsur.omega = eqsur.temp' * eqsur.temp /size(X_GM,1); %(size(X_GM,1)-size(X_GM,2));
eqsur.covar = inv(X'*inv(kron(eqsur.omega, eye(20)))*X);
eqsur.sdbeta = sqrt(diag(eqsur.covar));
eqsur.tbeta = eqsur.beta ./ eqsur.sdbeta;
eqsur.se = sqrt(diag(eqsur.omega));
%print results
fprintf('SUR estimates\n');
for jj = 1:5 % Loop over firms
    fprintf('\n\n\n')
    disp('                coef        sd    t-stat')
    for ii=1:3 %Loop over coefficients
        fprintf('%s%10.4f%10.4f%10.4f\n',label(ii), ...
            eqsur.beta(ii+(jj-1)*3),eqsur.sdbeta(ii+(jj-1)*3), ...
            eqsur.tbeta(ii+(jj-1)*3))
    end
    fprintf('Standard Error is %10.4f\n',eqsur.se(jj));
end
```

```
SUR estimates



                coef        sd    t-stat
C -168.1134   84.9017   -1.9801
F    0.1219    0.0204    5.9700
C    0.3822    0.0321   11.9109
Standard Error is    84.9836



                coef        sd    t-stat
C    0.9980   11.5661    0.0863
F    0.0689    0.0170    4.0473
C    0.3084    0.0260   11.8766
Standard Error is    12.3789
```

```
               coef        sd    t-stat
C  -21.1374   24.3479   -0.8681
F    0.0371    0.0115    3.2327
C    0.1287    0.0212    6.0728
Standard Error is    26.5467
```

```
               coef        sd    t-stat
C    1.4075    5.9944    0.2348
F    0.0564    0.0106    5.3193
C    0.0429    0.0382    1.1233
Standard Error is     9.7420
```

```
               coef        sd    t-stat
C   62.2563   93.0441    0.6691
F    0.1214    0.0451    2.6948
C    0.3691    0.1070    3.4494
Standard Error is    90.4117
```

SUR in LeSage toolbox

```
y(1).eq = Y_GM;
y(2).eq = Y_GE;
y(3).eq = Y_CH;
y(4).eq = Y_WE;
y(5).eq = Y_US;
XX(1).eq = X_GM;
XX(2).eq = X_GE;
XX(3).eq = X_CH;
XX(4).eq = X_WE;
XX(5).eq = X_US;
neqs=5;
sur_result=sur(neqs,y,XX);
prt(sur_result)
```

```
Seemingly Unrelated Regression -- Equation   1
System R-sqr   =    0.8694
R-squared      =    0.9207
Rbar-squared   =    0.9113
```

```
sigma^2        = 458183.2995
Durbin-Watson  =    0.0400
Nobs, Nvars    =     20,     3
***************************************************************
Variable          Coefficient      t-statistic    t-probability
variable   1      -168.113426        -1.980094         0.064116
variable   2         0.121906         5.969973         0.000015
variable   3         0.382167        11.910936         0.000000


Seemingly Unrelated Regression -- Equation   2
System R-sqr   =    0.8694
R-squared      =    0.9116
Rbar-squared   =    0.9012
sigma^2        = 8879.1368
Durbin-Watson  =    0.0310
Nobs, Nvars    =     20,     3
***************************************************************
Variable          Coefficient      t-statistic    t-probability
variable   1         0.997999         0.086286         0.932247
variable   2         0.068861         4.047270         0.000837
variable   3         0.308388        11.876603         0.000000


Seemingly Unrelated Regression -- Equation   3
System R-sqr   =    0.8694
R-squared      =    0.6857
Rbar-squared   =    0.6488
sigma^2        = 11785.8684
Durbin-Watson  =    0.0202
Nobs, Nvars    =     20,     3
***************************************************************
Variable          Coefficient      t-statistic    t-probability
variable   1       -21.137397        -0.868140         0.397408
variable   2         0.037053         3.232726         0.004891
variable   3         0.128687         6.072805         0.000012


Seemingly Unrelated Regression -- Equation   4
System R-sqr   =    0.8694
R-squared      =    0.7264
Rbar-squared   =    0.6943
```

```
sigma^2        = 2042.8631
Durbin-Watson  =    0.0323
Nobs, Nvars    =     20,     3
***************************************************************
Variable          Coefficient      t-statistic    t-probability
variable   1         1.407487         0.234802         0.817168
variable   2         0.056356         5.319333         0.000056
variable   3         0.042902         1.123296         0.276925


Seemingly Unrelated Regression -- Equation   5
System R-sqr   =    0.8694
R-squared      =    0.4528
Rbar-squared   =    0.3884
sigma^2        = 173504.8346
Durbin-Watson  =    0.0103
Nobs, Nvars    =     20,     3
***************************************************************
Variable          Coefficient      t-statistic    t-probability
variable   1        62.256312         0.669105         0.512413
variable   2         0.121402         2.694815         0.015340
variable   3         0.369111         3.449403         0.003062

Cross-equation sig(i,j) estimates
equation      eq 1      eq 2      eq 3      eq 4      eq 5
eq 1     7222.2204 -315.6107  601.6316  129.7644 -2446.3171
eq 2     -315.6107  153.2369    3.1478   16.6475  414.5298
eq 3      601.6316    3.1478  704.7290  201.4385 1298.6953
eq 4      129.7644   16.6475  201.4385   94.9067  613.9925
eq 5     -2446.3171  414.5298 1298.6953  613.9925 8174.2798


Cross-equation correlations
equation      eq 1      eq 2      eq 3      eq 4      eq 5
eq 1        1.0000   -0.3000    0.2667    0.1567   -0.3184
eq 2       -0.3000    1.0000    0.0096    0.1380    0.3704
eq 3        0.2667    0.0096    1.0000    0.7789    0.5411
eq 4        0.1567    0.1380    0.7789    1.0000    0.6971
eq 5       -0.3184    0.3704    0.5411    0.6971    1.0000
```

6.2 Exercise – Using Matlab to estimate a simultaneous equation systems

Consider the demand-supply model

$$\begin{aligned} q_t &= \beta_{11} + \beta_{21} x_{t2} + \beta_{31} x_{t2} + \gamma_{21} p_t + u_{t1} & (1) \\ q_t &= \beta_{12} + \beta_{42} x_{t4} + \beta_{52} x_{t5} + \gamma_{22} p_t + u_{t2}, & (2) \end{aligned}$$

where q_t is the log of quantity, p_t is the log of price, x_{t2} is the log of income, x_{t3} is a dummy variable that accounts for demand shifts x_{t4} and x_{t5} are input prices. Thus equations (1) and (2) are demand and supply functions respectively. 120 observations generated by this model are in the file `demand-supply.csv`

1. Comment on the identification of the system. Why can the system not be estimated using equation by equation OLS. For each of the estimates below produce estimates, standard errors and t-statistics of each coefficient. Also produce standard errors for each equation.
2. Estimate the system equation by equation using OLS.
3. Estimate the system equation by equation using 2SLS. Compare the results with the OLS estimates.
4. Set up the matrices of included variables, exogenous variables required to do system estimation.
5. Do OLS estimation using the stacked system and compare results with the equation by equation estimates.
6. Do 2SLS estimation using the stacked system and compare results with the equation by equation estimates.
7. Do 3SLS estimation using the stacked system and compare results with the 2SLS estimates.
8. Comment on the identification of the system.
9. How can the method be generalised to estimate other GMM estimators? Estimate the optimum GMM estimator for the system and compare your results with the previous estimators.

7 User written functions in MATLAB

One of the most useful facilities in MATLAB is the facility to write ones own functions and use them in the same way as a native MATLAB functions. We are already familiar with m-files which contain lists of MATLAB instructions. Such files are known as script files and allow us to do repeat an analysis without having to retype all the instructions. Suppose we wish to write a function that estimates the density function of a normal distribution,

$$\frac{1}{\sqrt{2\pi}\ \sigma} \exp - \frac{(x-\mu)^2}{2\sigma^2}$$

, we have the following on a file normdensity.m

```
function f = normdensity(z, mu, sigma);
% Calculates the Density Function of the Normal Distribution
% with mean mu
% and standard deviation sigma
% at a point z
% sigma must be a positive non-zero real number
if sigma  <= 0
    fprintf('Invalid input\n');
    f = NaN;
else
    f = (1/(sqrt(2*pi)*sigma))*exp(-(z-mu)^2/(2*sigma^2));
end
```

Note the following

1. The file starts with the keyword function. This is to indicate that this m-file is a function definition.
2. In the first line the f indicates that the value of f when the file has been "run" is the value that will be returned.
3. The function is called with normdensity(z, mu, sigma) where z mu and sigma are given values in calling the function.
4. The commented lines immediately after the first line are a help system for the function
5. All variables within a function are local to the function. Thus if there is a variable within a function called x and one in the program with the same name the one in the program is used when the program is in operation. Once the program has been run the value in the program is forgotten and the value outside the program is used.

The use of the function normdensity can be demonstrated as follows –

Get help for normdensity function.

```
help normdensity
```

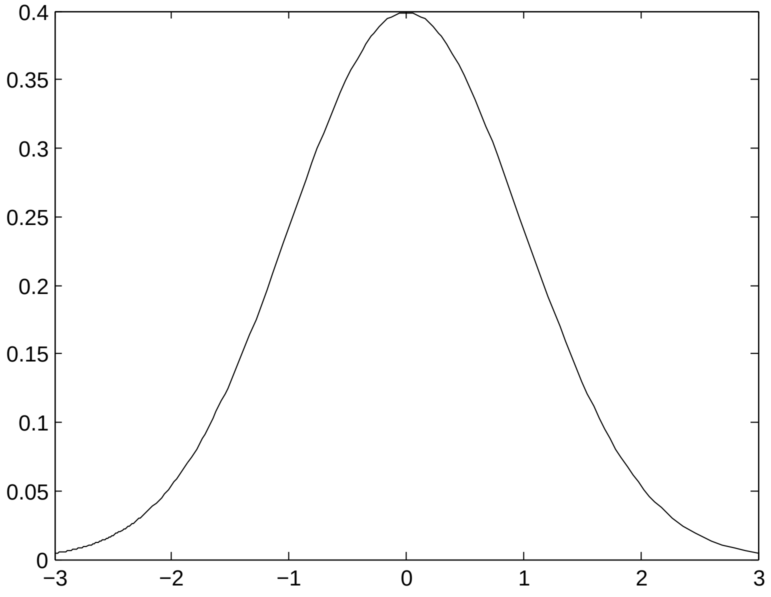
```
  Calculates the Density Function of the Normal Distribution
  with mean mu
  and standard deviation sigma
  at a point z
  sigma must be a positive non-zero real number
```

Evaluate Standard Normal density function at zero

```
normdensity(0,0,1)

ans =
    0.3989
```

Plot standard normal density function

```
fplot('normdensity(x,0,1)',[-3 3])
```



8 The LeSage Econometric Toolbox

If you are accustomed to using one of the many packages that deal specifically with econometrics you may think that MATLAB takes a long time to do simple things. It is also clear that many or the more difficult tasks are often easier in MATLAB than in these packages. MATLAB is less of a "'black box" than many of the other programs. One must really learn and understand the algebra before one can use MATLAB for econometrics. One also knows exactly what one is doing when one has written the routines in MATLAB.

The big problem is the lack of elementary econometric facilities in MATLAB. The LeSage MATLAB econometric package adds many of the required functions. It contains about 300 functions, utilities and demonstration programs. A list is included in Appendix A to this note. Full details are available in the toolbox manual which is available at `http://www.spatial-econometrics.com/`. The toolbox is designed to produce documentation, example programs, printed and graphical output across a wide range of econometric procedures.

Availability on Public Access Computers

The toolbox has been added to the MATLAB system on all Public Access computers on the TCD network. The functions are available in the same way as the ordinary MATLAB functions. For example, if y is a $n \times 1$ vector and X is a $n \times k$ matrix, the instruction

```
result = ols(y,X)
```

calculates the regression of y on X and various related statistics. The instruction

```
prt_reg(result)
```

produces a summary of the result.

Help is available in the command line in MATLAB. Help on the ols function can be found as follows

```
>> help ols
  PURPOSE: least-squares regression
 ---------------------------------------------------
  USAGE: results = ols(y,x)
  where: y = dependent variable vector    (nobs x 1)
         x = independent variables matrix (nobs x nvar)
 ---------------------------------------------------
  RETURNS: a structure
         results.meth  = 'ols'
         results.beta  = bhat     (nvar x 1)
         results.tstat = t-stats  (nvar x 1)
         results.bstd  = std deviations for bhat (nvar x 1)
         results.yhat  = yhat     (nobs x 1)
         results.resid = residuals (nobs x 1)
         results.sige  = e'*e/(n-k)   scalar
         results.rsqr  = rsquared     scalar
         results.rbar  = rbar-squared scalar
         results.dw    = Durbin-Watson Statistic
         results.nobs  = nobs
         results.nvar  = nvars
         results.y     = y data vector (nobs x 1)
         results.bint  = (nvar x 2 ) vector with 95% confidence intervals on beta
 ---------------------------------------------------
  SEE ALSO: prt(results), plt(results)
 ---------------------------------------------------

    Overloaded functions or methods (ones with the same name in other directories)
       help localmod/ols.m
```

After running the `ols` function a structure containing the results is available. The variable `results.beta` contains the estimated β-coefficients, `results.tstat` their t-statistics, `results.bint` the 95% confidence intervals for the estimates and similar for the other variables defined. Each estimation command produces its results in a similar structure. To see how to print a summary of these results

```
>> help prt_reg
PURPOSE: Prints output using regression results structures
 ---------------------------------------------------
  USAGE: prt_reg(results,vnames,fid)
  Where: results = a structure returned by a regression
         vnames  = an optional vector of variable names
         fid     = optional file-id for printing results to a file
                   (defaults to the MATLAB command window)
 ---------------------------------------------------
   NOTES: e.g. vnames = strvcat('y','const','x1','x2');
          e.g. fid = fopen('ols.out','wr');
   use prt_reg(results,[],fid) to print to a file with no vnames
  --------------------------------------------------
   RETURNS: nothing, just prints the regression results
  --------------------------------------------------
  SEE ALSO: prt, plt
 ---------------------------------------------------
```

Thus to display the results of the previous regression on the screen in MATLAB one would enter[3]

```
prt_reg(result)
```

[3]result in this context is the name of a MATLAB variable and one could substitute for `result` any valid MATLAB variable name.

Availability on other PCs

The LeSage toolbox is available for download free on the internet. The only requirement on the package web site is that "Anyone is free to use these routines, no attribution (or blame) need be placed on the author/authors." The econometrics package is not available by default when MATLAB is installed on a PC. It may be downloaded from `http://www.spatial-econometrics.com/`. The toolbox is provided as a zipped file which can be unzipped to the MATLAB toolbox directory on your PC (`C:\ProgramFiles\MATLAB704\toolbox` or my PC - something similar on yours). This should create a subdirectory econometrics in this toolbox directory This econometrics directory will contain a large number of subdirectories containing the various econometric functions. When you next start MATLAB you can access the functions by adding to the path that MATLAB uses to search for functions. You can do the when you next start MATLAB by —File—Set Path— selecting the Add with sub-folders button and navigating to and selecting the econometrics folder. If you select save after entering the directory the functions will be available each time you start MATLAB. If you have the required permissions you can also access the toolbox from the IIS server.

The toolbox provides full source code for each function. Thus if no function provides the capabilities that you require it may be possible the amend the function and add the required functionality. If you do such work you should consider submitting you program for inclusion in a future version of the toolbox. By collaborating in this way you are helping to ensure the future of the project

The programs in the toolbox are examples of good programming practice and have good comments. If you are starting some serious programming in MATLAB you could learn a lot about programming by reading these programs.

Sample run from the LeSage toolbox

To illustrate the use of the LeSage toolbox I set out below the output of the demonstration program `demo_ reg.m`. This program illustrates many of the various univariate estimation procedures available in the toolbox.

```
% PURPOSE: demo using most all regression functions
%
%       ols,hwhite,nwest,ridge,theil,tsls,logit,probit,tobit,robust
%---------------------------------------------------
% USAGE: demo_all
%---------------------------------------------------
clear all;

rand('seed',10);
n = 100; k=3;
```

```
xtmp = randn(n,k-1);
tt = 1:n;
ttp = tt';

e = randn(n,1).*ttp; % heteroscedastic error term
%e = randn(n,1);     % homoscedastic error term
b = ones(k,1);
iota = ones(n,1);
x = [iota xtmp];
% generate y-data
y = x*b + e;

vnames=strvcat('yvar','iota','x1','x2');

%  * * * * * * *  demo ols regression
reso = ols(y,x);
prt(reso,vnames);

%  * * * * * * *  demo hwhite regression
res = hwhite(y,x);
prt(res,vnames);

%  * * * * * * *  demo nwest regression
nlag=2;
res = nwest(y,x,nlag);
prt(res,vnames);

%  * * * * * * *  demo ridge regresson
rres = ridge(y,x);
prt(rres,vnames);

% * * * * * * *  demo logit regression
n = 24;
y = zeros(n,1);
y(1:14,1) = ones(14,1);
%       (data from Spector and Mazzeo, 1980)
xdata = [21 24 25 26 28 31 33 34 35 37 43 49 ...
         51 55 25 29 43 44 46 46 51 55 56 58];

iota = ones(n,1);
x = [iota xdata'];
```

```
vnames=strvcat('days','iota','response');

res = logit(y,x);
prt(res,vnames);

% * * * * * * *  demo probit regression
n = 32; k=4;
y = zeros(n,1); % grade variable
y(5,1)  = 1;
y(10,1) = 1;
y(14,1) = 1;
y(20,1) = 1;
y(22,1) = 1;
y(25,1) = 1;
y(25:27,1) = ones(3,1);
y(29,1) = 1;
y(30,1) = 1;
y(32,1) = 1;

x = zeros(n,k);

x(1:n,1) = ones(n,1);      % intercept
x(19:32,2) = ones(n-18,1); % psi variable
tuce = [20 22 24 12 21 17 17 21 25 29 20 23 23 25 26 19 ...
        25 19 23 25 22 28 14 26 24 27 17 24 21 23 21 19];

x(1:n,3) = tuce';

gpa = [2.66 2.89 3.28 2.92 4.00 2.86 2.76 2.87 3.03 3.92 ...
       2.63 3.32 3.57 3.26 3.53 2.74 2.75 2.83 3.12 3.16 ...
       2.06 3.62 2.89 3.51 3.54 2.83 3.39 2.67 3.65 4.00 ...
       3.10 2.39];

x(1:n,4) = gpa';

vnames=strvcat('grade','iota','psi','tuce','gpa');

resp = probit(y,x);

prt(resp,vnames);
% results reported in Green (1997, chapter 19)
% b = [-7.452, 1.426, 0.052, 1.626 ]
%  * * * * * * *  demo theil-goldberger regression
% generate a data set
nobs = 100;
nvar = 5;
beta = ones(nvar,1);
beta(1,1) = -2.0;

xmat = randn(nobs,nvar-1);
x = [ones(nobs,1) xmat];
evec = randn(nobs,1);

y = x*beta + evec*10.0;

Vnames = strvcat('y','const','x1','x2','x3','x4');

% set up prior
rvec = [-1.0    % prior means for the coefficients
         1.0
         2.0
         2.0
         1.0];

rmat = eye(nvar);
bv = 10000.0;

% umat1 = loose prior

umat1 = eye(nvar)*bv; % initialize prior variance as diffuse

for i=1:nvar;
umat1(i,i) = 1.0;     % overwrite diffuse priors with informative prior
end;

lres = theil(y,x,rvec,rmat,umat1);

prt(lres,Vnames);

%  * * * * * * *  demo two-stage least-squares regression

nobs = 200;
```

```
x1 = randn(nobs,1);
x2 = randn(nobs,1);
b1 = 1.0;
b2 = 1.0;
iota = ones(nobs,1);

y1 = zeros(nobs,1);
y2 = zeros(nobs,1);
evec = randn(nobs,1);

% create simultaneously determined variables y1,y2
for i=1:nobs;
y1(i,1) = iota(i,1)*1.0 + x1(i,1)*b1 + evec(i,1);
y2(i,1) = iota(i,1)*1.0 + y1(i,1)*1.0 + x2(i,1)*b2 + evec(i,1);
end;

vname2 = ['y2-eqn  ',
          'y1 var  ',
          'constant',
          'x2 var  '];

% use all exogenous in the system as instruments
xall = [iota x1 x2];

% do tsls regression
result2 = tsls(y2,y1,[iota x2],xall);
prt(result2,vname2);


% * * * * * * * demo robust regression

% generate data with 2 outliers

nobs = 100;
nvar = 3;

vnames = strvcat('y-variable','constant','x1','x2');

x = randn(nobs,nvar);

x(:,1) = ones(nobs,1);
beta = ones(nvar,1);
```

```
evec = randn(nobs,1);

y = x*beta + evec;

% put in 2 outliers
y(75,1) = 10.0;
y(90,1) = -10.0;

% get weighting parameter from OLS
% (of course you're free to do differently)
reso = ols(y,x);
sige = reso.sige;

% set up storage for bhat results
bsave = zeros(nvar,5);
bsave(:,1) = ones(nvar,1);

% loop over all methods producing estimates
for i=1:4;

wfunc = i;
wparm = 2*sige; % set weight to 2 sigma

res = robust(y,x,wfunc,wparm);

bsave(:,i+1) = res.beta;

end;
% column and row-names for mprint function
in.cnames = strvcat('Truth','Huber t','Ramsay','Andrews','Tukey');
in.rnames = strvcat('Parameter','constant','b1','b2');
fprintf(1,'Comparison of alternative robust estimators \n');
mprint(bsave,in);

res = robust(y,x,4,2);

prt(res,vnames);

%  * * * * * * *  demo regresson with t-distributed errors
res = olst(y,x);
prt(res,vnames);
```

```
%  * * * * * * *  demo lad regression
res = lad(y,x);
prt(res,vnames);

% * * * * * * * demo tobit regression
n=100; k=5;
x = randn(n,k);
x(:,1) = ones(n,1);
beta = ones(k,1)*0.5;
y = x*beta + randn(n,1);

% now censor the data
for i=1:n
 if y(i,1) < 0
 y(i,1) = 0.0;
 end;
end;

resp = tobit(y,x);

vnames = ['y     ',
          'iota  ',
          'x1var ',
          'x2var ',
          'x3var ',
          'x4var '];

prt(resp,vnames);


% * * * * * * * demo thsls regression

clear all;

nobs = 100;
neqs = 3;

x1 = randn(nobs,1);
x2 = randn(nobs,1);
x3 = randn(nobs,1);
b1 = 1.0;
b2 = 1.0;
```

```
b3 = 1.0;
iota = ones(nobs,1);

y1 = zeros(nobs,1);
y2 = zeros(nobs,1);
y3 = zeros(nobs,1);
evec = randn(nobs,3);
evec(:,2) = evec(:,3) + randn(nobs,1); % create cross-eqs corr

% create simultaneously determined variables y1,y2
for i=1:nobs;
y1(i,1) = iota(i,1)*10.0 + x1(i,1)*b1 + evec(i,1);
y2(i,1) = iota(i,1)*10.0 + y1(i,1)*1.0 + x2(i,1)*b2 + evec(i,2);
y3(i,1) = iota(i,1)*10.0 + y2(i,1)*1.0 + x2(i,1)*b2 + x3(i,1)*b3 + evec(i,3);
end;


vname1 = ['y1-LHS  ',
          'constant',
          'x1 var  '];

vname2 = ['y2-LHS  ',
          'y1 var  ',
          'constant',
          'x2 var  '];

vname3 = ['y3-LHS  ',
          'y2 var  ',
          'constant',
          'x2 var  ',
          'x3 var  '];


% set up a structure for y containing y's for each eqn
y(1).eq = y1;
y(2).eq = y2;
y(3).eq = y3;

% set up a structure for Y (RHS endogenous) for each eqn
Y(1).eq = [];
Y(2).eq = [y1];
Y(3).eq = [y2];
```

```
% set up a structure fo X (exogenous) in each eqn
X(1).eq = [iota x1];
X(2).eq = [iota x2];
X(3).eq = [iota x2 x3];

% do thsls regression

result = thsls(neqs,y,Y,X);

vname = [vname1
         vname2
         vname3];

prt(result,vname);

% * * * * * * * demo olsc, olsar1 regression

% generate a model with 1st order serial correlation
n = 200;
k = 3;
tt = 1:n;
evec = randn(n,1);
xmat = randn(n,k);
xmat(:,1) = ones(n,1);
beta = ones(k,1);
beta(1,1) = 10.0; % constant term
y = zeros(n,1);
u = zeros(n,1);

for i=2:n;
 u(i,1) = 0.4*u(i-1,1) + evec(i,1);
 y(i,1) = xmat(i,:)*beta + u(i,1);
end;

% truncate 1st 100 observations for startup
yt = y(101:n,1);
xt = xmat(101:n,:);
n = n-100; % reset n to reflect truncation

Vnames = ['y    ',
          'cterm',
          'x2   ',
          'x3   '];


% do Cochrane-Orcutt ar1 regression
result = olsc(yt,xt);
prt(result,Vnames);

% do maximum likelihood ar1 regression
result2 = olsar1(yt,xt);
prt(result2,Vnames);



% * * * * * * * demo switch_em, hmarkov_em regressions

clear all;

% generate data from switching regression model
nobs = 100; n1 = 3; n2 = 3; n3 = 3;
b1 = ones(n1,1); b2 = ones(n2,1)*5; b3 = ones(n3,1);
sig1 = 1; sig2 = 1;
randn('seed',201010);
x1 = randn(nobs,n1); x2 = randn(nobs,n2); x3 = randn(nobs,n3);
ytruth = zeros(nobs,1);
for i=1:nobs;
 if x3(i,:)*b3 <= 0
  y(i,1) = x1(i,:)*b1 + randn(1,1);
  ytruth(i,1) = 0;
 else
  y(i,1) = x2(i,:)*b2 + randn(1,1);
  ytruth(i,1) = 1;
 end;
end;

result = switch_em(y,x1,x2,x3,b1,b2,b3);

vnames1 = strvcat('y1','x1_1','x1_2','x1_3');
vnames2 = strvcat('y2','x2_1','x2_2','x2_3');
vnames3 = strvcat('x3_1','x3_2','x3_3');
vnames = [vnames1
          vnames2
          vnames3];

prt(result,vnames);

Ordinary Least-squares Estimates
Dependent Variable =             yvar
R-squared      =    0.0018
Rbar-squared   =   -0.0188
sigma^2        = 3075.1129
Durbin-Watson  =    1.9735
Nobs, Nvars    =    100,     3
***************************************************************
Variable      Coefficient      t-statistic    t-probability
iota            -1.899455        -0.338977         0.735360
x1              -2.301110        -0.358531         0.720725
x2              -1.298278        -0.220027         0.826312


White Heteroscedastic Consistent Estimates
Dependent Variable =             yvar
R-squared      =    0.0018
Rbar-squared   =   -0.0188
sigma^2        = 3075.1129
Durbin-Watson  =    1.9735
Nobs, Nvars    =    100,     3
***************************************************************
Variable      Coefficient      t-statistic    t-probability
iota            -1.899455        -0.322516         0.747756
x1              -2.301110        -0.390648         0.696914
x2              -1.298278        -0.176022         0.860644


Newey-West hetero/serial Consistent Estimates
Dependent Variable =             yvar
R-squared      =    0.0018
Rbar-squared   =   -0.0188
sigma^2        = 3075.1129
Durbin-Watson  =    1.9735
Nobs, Nvars    =    100,     3
***************************************************************
Variable      Coefficient      t-statistic    t-probability
iota            -1.899455        -0.343861         0.731695
x1              -2.301110        -0.349591         0.727403
```

```
x2              -1.298278       -0.189757         0.849896


Ridge Regression Estimates
Dependent Variable =             yvar
R-squared      =   -0.0007
Rbar-squared   =   -0.0213
sigma^2        = 3082.6476
Durbin-Watson  =    1.9909
Ridge theta    =        10.253908
Nobs, Nvars    =    100,     3
***************************************************************
Variable      Coefficient      t-statistic    t-probability
iota            -0.164038        -0.099106         0.921259
x1              -0.211398        -0.110545         0.912205
x2              -0.089360        -0.051208         0.959265


Logit Maximum Likelihood Estimates
Dependent Variable =         days
McFadden R-squared     =    0.1476
Estrella R-squared     =    0.1951
LR-ratio, 2*(Lu-Lr)    =    4.8131
LR p-value             =    0.0282
Log-Likelihood         =  -13.8941
# of iterations        =      6
Convergence criterion  =   5.2516501e-012
Nobs, Nvars            =     24,     2
# of 0's, # of 1's     =     10,    14
***************************************************************
Variable      Coefficient      t-statistic    t-probability
iota             3.819440         2.081230         0.049260
response        -0.086483        -2.001038         0.057876


Probit Maximum Likelihood Estimates
Dependent Variable =           grade
McFadden R-squared     =    0.3775
Estrella R-squared     =    0.4566
LR-ratio, 2*(Lu-Lr)    =   15.5459
LR p-value             =    0.0014
Log-Likelihood         =  -12.8188
```

```
# of iterations        =      7
Convergence criterion  =   2.1719878e-010
Nobs, Nvars            =     32,     4
# of 0's, # of 1's     =     21,    11
***************************************************************
Variable      Coefficient      t-statistic    t-probability
iota            -7.452320        -2.931131         0.006656
psi              1.426332         2.397045         0.023445
tuce             0.051729         0.616626         0.542463
gpa              1.625810         2.343063         0.026459


Theil-Goldberger Regression Estimates
Dependent Variable =           y
R-squared      =    0.0459
Rbar-squared   =    0.0057
sigma^2        =  103.8523
Durbin-Watson  =    2.1050
Nobs, Nvars    =    100,     5
***************************************************************
Variable       Prior Mean    Std Deviation
const           -1.000000        1.000000
x1               1.000000        1.000000
x2               2.000000        1.000000
x3               2.000000        1.000000
x4               1.000000        1.000000

***************************************************************
      Posterior Estimates
Variable      Coefficient      t-statistic    t-probability
const           -1.643936        -2.287731         0.024371
x1               0.591000         0.815968         0.416559
x2               2.176380         2.959987         0.003884
x3               1.674902         2.298068         0.023751
x4               0.629662         0.809268         0.420383


Two Stage Least-squares Regression Estimates
Dependent Variable =         y2-eqn
R-squared      =    0.7577
Rbar-squared   =    0.7552
sigma^2        =    1.4553
```

```
Durbin-Watson  =    1.7741
Nobs, Nvars    =    200,     3
***************************************************************
Variable      Coefficient      t-statistic    t-probability
y1 var           0.849977         8.010105         0.000000
constant         1.192429         8.683790         0.000000
x2 var           0.989913        12.700675         0.000000

Comparison of alternative robust estimators
Parameter      Truth    Huber t     Ramsay    Andrews      Tukey
constant      1.0000     0.9627     1.0288     0.9558     0.9159
b1            1.0000     1.0588     1.0498     1.0587     1.1143
b2            1.0000     0.8019     0.8862     0.8090     0.9775


Robust Regression Estimates
Dependent Variable =       y-variable
R-squared      =    0.3012
Rbar-squared   =    0.2868
Weighting meth =            tukey
Weight param   =    2.0000
sigma^2        =    3.8269
Durbin-Watson  =    1.7969
Nobs, Nvars    =    100,     3
# iterations   =     19
converg crit   =   8.0813581e-006
***************************************************************
Variable        Coefficient      t-statistic    t-probability
constant           0.939790         3.792787         0.000259
x1                 1.093821         5.003626         0.000003
x2                 1.062278         4.701951         0.000009


Regression with t-distributed errors
Dependent Variable =       y-variable
R-squared      =    0.3096
Rbar-squared   =    0.2953
sigma^2        =    3.6678
Durbin-Watson  =    1.7974
Nobs, Nvars    =    100,     3
# iterations   =     13
converg crit   =   2.5709227e-009
```

```
***************************************************************
Variable        Coefficient      t-statistic    t-probability
constant           0.921545         2.331888         0.021775
x1                 1.106885         2.969556         0.003758
x2                 0.981229         2.540883         0.012643


Least-Absolute Deviation Estimates
Dependent Variable =       y-variable
R-squared      =    0.3126
Rbar-squared   =    0.2984
sigma^2        =    3.7647
Durbin-Watson  =    1.7916
Nobs, Nvars    =    100,     3
# iterations   =     37
convergence    =   8.8817842e-016
***************************************************************
Variable        Coefficient      t-statistic    t-probability
constant           0.981256       131.486643         0.000000
x1                 1.071320       161.392625         0.000000
x2                 0.942192       267.374908         0.000000


Tobit Regression Estimates
Dependent Variable =           y
R-squared      =    0.9905
Rbar-squared   =    0.9901
sigma^2        =    1.4500
Log-Likelihood =       -128.86295
# iterations   =     11
optimization   =   bfgs
Nobs, Nvars    =    100,     5
# of censored  =     32
time (in secs) =       0.2
***************************************************************
gradient at solution
Variable         Gradient
iota           0.00023491
x1var         -0.00032300
x2var         -0.00027021
x3var          0.00025956
x4var         -0.00006834
```

```
sigma         0.00005784

Variable      Coefficient      t-statistic    t-probability
iota             0.524686         3.558525         0.000584
x1var            0.712329         5.060812         0.000002
x2var            0.557483         4.419124         0.000026
x3var            0.456688         3.354569         0.001143
x4var            0.567654         4.046847         0.000106


Three Stage Least-squares Estimates -- Equation   1
Dependent Variable =         y1-LHS
R-squared      =    0.5307
Rbar-squared   =    0.5259
sigma^2        =    0.8239
Durbin-Watson  =    1.8589
Nobs, Nvars    =    100,     2
***************************************************************
Variable      Coefficient      t-statistic    t-probability
constant         9.919085       108.989267         0.000000
x1 var           1.063664        10.642418         0.000000


Three Stage Least-squares Estimates -- Equation   2
Dependent Variable =         y2-LHS
R-squared      =    0.6531
Rbar-squared   =    0.6460
sigma^2        =    2.1255
Durbin-Watson  =    2.2631
Nobs, Nvars    =    100,     3
***************************************************************
Variable      Coefficient      t-statistic    t-probability
y1 var           1.271276         8.565909         0.000000
constant         7.252903         4.869941         0.000004
x2 var           1.016608         7.700645         0.000000


Three Stage Least-squares Estimates -- Equation   3
Dependent Variable =         y3-LHS
R-squared      =    0.9454
Rbar-squared   =    0.9436
sigma^2        =    0.7704
```

```
Durbin-Watson  =    2.2675
Nobs, Nvars    =    100,     4
***************************************************************
Variable      Coefficient      t-statistic    t-probability
y2 var           1.072609        15.286953         0.000000
constant         8.513420         6.070655         0.000000
x2 var           0.884799         7.971522         0.000000
x3 var           1.029391        18.715601         0.000000

Cross-equation sig(i,j) estimates
equation    y1-LHS    y2-LHS    y3-LHS
y1-LHS      0.8239   -0.1371    0.0736
y2-LHS     -0.1371    2.1238    0.9340
y3-LHS      0.0736    0.9340    0.7626


Cross-equation correlations
equation    y1-LHS    y2-LHS    y3-LHS
y1-LHS      1.0000   -0.1036    0.0928
y2-LHS     -0.1036    1.0000    0.7339
y3-LHS      0.0928    0.7339    1.0000


Cochrane-Orcutt serial correlation Estimates
Dependent Variable =            y
R-squared       =    0.6751
Rbar-squared    =    0.6683
sigma^2         =    1.0163
Durbin-Watson   =    2.0564
Rho estimate    =    0.4405
Rho t-statistic =    4.8569
Rho probability =    0.0000
Nobs, Nvars     =     99,     3
***************************************************************
Iteration information
       rho value      convergence    iteration
        0.440309         0.440309            1
        0.440463         0.000154            2
        0.440464         0.000000            3

***************************************************************
Variable      Coefficient      t-statistic    t-probability
```

```
cterm          10.109079       55.582539        0.000000
x2              1.008878       10.021495        0.000000
x3              1.117656       11.414945        0.000000


Maximum likelihood ar1 serial correlation Estimates
Dependent Variable =           y
R-squared       =    0.7012
Rbar-squared    =    0.6950
sigma^2         =    1.0124
Durbin-Watson   =    2.0425
Rho estimate    =    0.4387
Rho t-statistic =    4.8330
Rho probability =    0.0000
Nobs, Nvars     =    100,     3
Iterations      =      4
Log Likelihood  =       -229.46078
Time (in secs)  =       0.0
***************************************************************
Variable      Coefficient      t-statistic    t-probability
cterm          10.131625       56.672826        0.000000
x2              1.009420       10.039487        0.000000
x3              1.125448       11.566906        0.000000


EM Estimates - Switching Regression model
Regime 1 equation
Dependent Variable =            y1
R-squared      =    0.9377
sigma^2        =    0.8997
Nobs, Nvars    =    100,     3
***************************************
Variable      Coefficient      t-statistic    t-probability
x1_1            1.182283        8.051477        0.000000
x1_2            0.998518        6.233840        0.000000
x1_3            1.038357        7.625493        0.000000


Regime 2 equation
Dependent Variable =            y2
R-squared      =    0.9997
sigma^2        =    1.0544
```

```
Nobs, Nvars    =    100,     3
***************************************
Variable      Coefficient      t-statistic    t-probability
x2_1             5.164178        25.388701         0.000000
x2_2             4.763510        29.745264         0.000000
x2_3             4.909741        30.189646         0.000000

Switching equation
Conv criterion =    0.00099191605
# iterations   =     43
# obs regime 1 =     54
# obs regime 2 =     46
log Likelihood =       -395.16724
Nobs, Nvars    =    100,     3
***************************************
Variable      Coefficient      t-statistic    t-probability
x3_1             1.027361        10.097151         0.000000
x3_2             1.061341        10.089601         0.000000
x3_3             0.998786        11.024366         0.000000
```

9 Maximum Likelihood Estimation using Numerical Techniques

In many cases of maximum likelihood estimation there is no analytic solution to the optimisation problem and one must use numerical techniques. This basic maximum likelihood algorithm is similar in many econometric packages. In most cases one works with the log-likelihood rather than the likelihood. Note that many packages contain a minimisation routine rather than a maximisation and thus one seeks to minimise the negative of the log-likelihood. The steps involved are as follows –

1. Write a MATLAB function to estimate the log-likelihood.
2. Load and process data.
3. Calculate an initial estimate of the parameters to be estimated. This will serve as starting values for the optimisation routine.
4. Check the defaults for the optimisation routine (e.g. maximum number of iterations, convergence criteria). If your initial attempt does not converge you may have to change these and/or use different starting values.
5. Call the optimisation routine. Optimisation routines are available in the MATLAB **optim** toolbox or in the Le Sage **econometrics** package. As the **optim** package

is an add-on which is not included in standard MATLAB I shall illustrate the routines using optimisation routines taken from the **econometrics package**

6. Print out results.

I shall illustrate the procedure by replicating the tobit analysis of tobacco expenditure on in Table 7.9 on page 237 of Verbeek (2008). The population is assumed to be censored at $y = 0$. Define

$$d = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{if } y \leq 0. \end{cases}$$

The log-likelihood can then be written

$$\sum_{i=1}^{N} \left[d_i(-\frac{1}{2}\ln 2\pi - \frac{1}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i\boldsymbol{\beta})^2) + (1-d_i)\ln\left(1 - \Phi\left(\frac{\boldsymbol{x\beta}}{\boldsymbol{\sigma}}\right)\right)\right]$$

1. Matlab program to calculate tobit log-likelihood

This is an amended version of a sample program included with the Le Sage **econometrics** package

```
function like =  to_liked(b,y,x);
% PURPOSE: evaluate tobit log-likelihood
%          to demonstrate optimization routines
%-----------------------------------------------------
% USAGE:    like = to_liked(b,y,x)
% where:     b = parameter vector (k x 1)
%            y = dependent variable vector (n x 1)
%            x = explanatory variables matrix (n x m)
%-----------------------------------------------------
% NOTE: this function returns a scalar equal to the
%       negative of the log likelihood function
%       or a scalar sum of the vector depending
%       on the value of the flag argument
%       k ~= m because we may have additional parameters
%           in addition to the m bhat's (e.g. sigma)
%-----------------------------------------------------
% error check
if nargin ~= 3,error('wrong # of arguments to to_like1'); end;
[m1 m2] = size(b);
if m1 == 1
 b = b';
end;
```

```
   h = .000001;             % avoid sigma = 0
   [m junk] = size(b);
   beta = b(1:m-1);         % pull out bhat
   sigma = max([b(m) h]);   % pull out sigma
   xb = x*beta;
   llf1 =   -0.5*log(2*pi) - 0.5*log(sigma^2) -((y-xb).^2)./(2*sigma^2); %amended
   xbs = xb./sigma; cdf = .5*(1+erf(xbs./sqrt(2))); %amended
   llf2 = log(h+(1-cdf));
   llf = (y > 0).*llf1 + (y <= 0).*llf2;
   like = -sum(llf); % scalar result
```

2. Load and process data

```
clear;
load cigarette
% set up variables
y = sharetob;
[nobs, dump] = size(tobacco) ;
X = [ones(nobs,1), age, nadults, nkids, nkids2, lnexp, age.*lnexp, nadults.*lnexp];
[nobs, k]=size(X);
```

3. Estimate starting values using OLS

```
beta0 = ( X'*X)\X'*y ;
sd = sqrt((y-X*beta0)'*(y-X*beta0))/(nobs-1);
parm = [beta0
    sd ];
```

4 .Set up dfp_min defaults - change maximum number of iterations

```
info.maxit = 1000;
```

5. Call optimisation routine

```
result2 = dfp_min('to_liked',parm,info,y,X);
```

Extract results and print

```
% Extract results
beta = result2.b;
sdbeta = sqrt(diag(inv(result2.hess)));
```

```
zstat = beta  ./ sdbeta;
%row names
rnames = 'b1';
for i=2:k;
bstring = ['b' num2str(i)];
rnames = strvcat(rnames,bstring);
end;
rnames = strvcat(rnames,'sigma');

disp('Estimates of tobit')
disp('           coef        sd    t-stat')
for ii=1:k+1
fprintf('%s%10.5f%10.5f%10.5f\n',rnames(ii,:),beta(ii),sdbeta(ii),zstat(ii))
end
```

```
Estimates of tobit
           coef        sd    t-stat
b1      0.58932   0.06228   9.46266
b2     -0.12572   0.02313  -5.43565
b3      0.01570   0.00000   0.00000
b4      0.00427   0.00132   3.23039
b5     -0.00997   0.00547  -1.82376
b6     -0.04438   0.00465  -9.55517
b7      0.00881   0.00170   5.17910
b8     -0.00062   0.00000   0.00000
sigma   0.04800   0.00020 237.12802
```

This example of numerical optimisation works well. You can check that you get a similar answer from the tobit function in any econometric package. In many real cases it will not be that easy. You will be using numerical routines because your econometric package does not have a function to do the required analysis. There are many pitfalls that lie in wait for the economist trying to do numerical optimisation. One should never be satisfied with just one run as above. I did check the results with Verbeek (2008) and another package and did have to do some amendment to the original programs. In all cases of non-linear optimisation you should

1. Ensure that the process has converged. The results from a model that has not converged are totally useless, no matter how good they look.

2. Is there a corresponding published analysis that you can duplicate. Failure to replicate may indicate a problem with the published data. Can you simulate a data set with the statistical properties of your real data set? Can you estimate correctly the model underlying the simulated data.

3. If you have problems getting your estimates to converge it may be worth while rescaling your data so that the means and standard deviations are similar.

4. Is your likelihood surface flat close to the optimum – Your model may be over-elaborate for your data.

5. The likelihood function may have multiple local maxima. Unless your mathematics tells you that thee is only one local maximum you should check that you have found the true optimum. Does an alternative set initial values lead to a different optimum.

10 Octave, Scilab and R

MATLAB is an excellent program and is widely used in finance, science and engineering. It has a very good user interface. There may be occasions when you do not have access to MATLAB and require urgent access. For security reasons, your employer may place various restrictions on the programs you can use on a work computer. If you want to use MATLAB you may need local management approval, IT management approval and purchase through a central purchasing unit. By the time you get MATLAB you may find that the need has passed. In such a case, you might consider Octave or Scilab which are two free programs with similar functionality to MATLAB.

10.1 Octave

Octave is largely compatible with MATLAB. Up to recently there were considerable problems running Octave on windows and it could be recommended only for those with considerable expertise in MS Windows and some knowledge of a Unix based systems. The new version 3 of Octave has solved these problems. The interface is different to the MATLAB interface but this should not lead to any great problems.

A program for base MATLAB will run in Octave with at most minor changes and, in all likelihood with none. The original drafts of these note were completed with Octave as I had no easy access to MATLAB at the time. Programs written in Octave may not run in base MATLAB as base Octave contains many functions similar to those in add-on MATLAB toolboxes. These make Octave a better package for econometrics than base MATLAB. Creel (2008) is a set of econometrics notes based with applications in Octave. Examples, data-sets and programs are available on the web with the notes.

It is claimed that the LeSage package runs in Octave. I have tested some of the packages and they are compatible but have not installed the entire toolbox.

10.2 Scilab

Scilab is another free matrix manipulation language available from `www.scilab.org`. Scilab has the same basic functionality as MATLAB but its syntax is a little different. The functionality of both language is so similar that anyone accustomed to programming in MATLAB should have no problems reading Scilab programs but Scilab programs will need editing before they could be used in MATLAB. Scilab contains a utility for translating MATLAB programs to Scilab. This utility works well. Campbell et al. (2006) is a good introduction to Scilab and contains a lot of tutorial material. There is also an econometrics toolbox for Scilab called GROCER. While this package is partly derived from the LeSage package it has a lot of extra features that might be useful.

One may well ask which is the best program. MATLAB is definitely the market leader in the field. It is very much embedded in the scientific/engineering fields with branches in Finance. It has applications in advanced macroeconomics and is a suitable tool for empirical research. Octave is very similar to MATLAB but has only recently made the transition to MS Windows. Octave has better facilities than base MATLAB. The combination of Scilab and GROCER makes a most interesting tool for economics. If I was working in a MATLAB environment where good support was available in-house I would not try anything else. If I wanted a program to run my MATLAB programs at home and did not want to go to the expense of acquiring a licence for basic MATLAB and tools I would try Octave first. If I was just interested in some private empirical research Scilab would be worth trying. Experience gained in programming Octave or Scilab would transfer easily to MATLAB.

10.3 R

Faced with these alternatives my personal choice has been R. R is "GNU S", a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. More information is available from The Comprehensive R Archive Network (CRAN) at `http://www.r-project.org/` or at one of its many mirror sites. Not only does R cover all aspects of statistics but it has most of the computational facilities of MATLAB. It is the package in which most academic statistical work is being completed. There is a large amount of free tutorial material available on CRAN and an increasing number of textbooks on R have been published in recent years.

If it can not be done in basic R then one is almost certain to find a solution in one of the 1600+ "official packages" on CRAN or the "unofficial packages" on other sites. R is regarded as having a steep learning curve but there are several graphical interfaces that facilitate the use of R. Summary information of the use of R in economics and finance can be seen on the Task views on the CRAN web site or on one of its many mirrors. Kleiber and Zeileis (2008) is a good starting point for econometrics in R.

References

Campbell, S. L., J.-P. Chancelier, and R. Nikoukhah (2006). *Modeling and Simulation in Scilab/Scicos*. Springer.

Creel, M. (2008). Econometrics. http://pareto.uab.es/mcreel/Econometrics/.

Green, W. H. (2000). *Econometric Analysis* (fourth ed.). Prentice Hall.

Green, W. H. (2008). *Econometric Analysis*. Pearson Prentice Hall.

Higham, D. J. and N. J. Higham (2005). *MATLAB Guide*. Society for Industrial and Applied Mathematics.

Kendrick, D. A., P. Mercado, and H. M. Amman (2006). *Computational Economics*. Princeton University Press.

Kleiber, C. and A. Zeileis (2008). *Applied Econometrics with R*. Springer.

LeSage, J. P. (1999, October). Applied econometrics using MATLAB.

Ljungqvist, L. and T. J. Sargent (2004). *Recursive Macroeconomic Theory* (Second edition ed.). Princeton University Press.

Marimon, R. and A. Scott (Eds.) (1999). *Computational Methods for the Study of Dynamic Economies*. Oxford University Press.

Miranda, M. J. and P. L. Fackler (2002). *Applied Computational Economics and Finance*. Princeton University Press.

Paolella, M. S. (2006). *Fundamental Probability: A Computational Approach*. Wiley.

Paolella, M. S. (2007). *Intermediate Probability: A Computational Approach*. Wiley.

Pratap, R. (2006). *Getting Started with MATLAB 7 : A Quick Introduction for Scientists and Engineers*. Oxford University Press.

Shone, R. (2002). *Economic Dynamics* (Second ed.). Cambridge.

Verbeek, M. (2008). *A Guide to modern Econometrics* (Third ed.). Wiley.

A Functions etc. in LeSage Econometrics Toolbox

A.1 Regression

The regression function library is in a subdirectory `regress`.

A.1.1 Programs

program	description
ar_ g	Gibbs sampling Bayesian autoregressive model
bma_ g	Gibbs sampling Bayesian model averaging
boxcox	Box-Cox regression with 1 parameter
boxcox2	Box-Cox regression with 2 parameters
egarchm	EGARCH(p,q)-in-Mean regression model
emhergott	EM estimates of Hamilton's markov switching model
garchs	garchs(1,1,1) model using non-central t-distribution
hmarkov_ em	Hamilton's markov switching model
hwhite	Halbert White's heteroscedastic consistent estimates
lad	least-absolute deviations regression
lm_test	LM-test for two regression models
logit	logit regression
mlogit	multinomial logit regression
nwest	Newey-West hetero/serial consistent estimates
ols	ordinary least-squares
ols_g	Gibbs sampling Bayesian linear model
olsar1	Maximum Likelihood for AR(1) errors ols model
olsc	Cochrane-Orcutt AR(1) errors ols model
olst	regression with t-distributed errors
probit	probit regression
probit_ g	Gibbs sampling Bayesian probit model
ridge	ridge regression
robust	iteratively reweighted least-squares
rtrace	ridge estimates vs parameters (plot)
sur	seemingly unrelated regressions
switch_ em	switching regime regression using EM-algorithm
theil	Theil-Goldberger mixed estimation
thsls	three-stage least-squares
tobit	tobit regression
tobit_ g	Gibbs sampling Bayesian tobit model
tsls	two-stage least-squares
waldf	Wald F-test

A.1.2 Demonstrations

program	description
ar_ gd	demonstration of Gibbs sampling ar_ g
bma_ gd	demonstrates Bayesian model averaging
boxcox_	demonstrates Box-Cox 1–parameter model
boxcox2_	demonstrates Box-Cox 2–parmaeter model
demo_ all	demos most regression functions
egarchm_ d	demos egarchm function
garchs_ d	demos garchs function
hmarkov_ emd	demos Hamilton's model
hmarkov_ emd2	another demo of Hamilton's model
hwhite_ d	H. White's hetero consistent estimates demo
lad_ d	demos lad regression
lm_ test_ d	demos lm_ test
logit_ d	demonstrates logit regression
mlogit_ d	demonstrates multinomial logit
nwest_ d	demonstrates Newey-West estimates
ols_ d	demonstrates ols regression
ols_ d2	Monte Carlo demo using ols regression
ols_ gd	demo of Gibbs sampling ols_ g
olsar1_ d	Max Like AR(1) errors model demo
olsc_ d	Cochrane-Orcutt demo
olst_ d	olst demo
probit_ d	probit regression demo
probit_ gd	demo of Gibbs sampling Bayesian probit model
ridge_ d	ridge regression demo
robust_ d	demonstrates robust regression
sur_ d	demonstrates sur using Grunfeld's data
switch_ emd	demonstrates switching regression
theil_ d	demonstrates theil-goldberger estimation
thsls_ d	three-stage least-squares demo
tobit_ d	tobit regression demo
tobit_ d2	tobit right-censoring demo
tobit_ gd	demo of Gibbs sampling Bayesian tobit model
tobit_ gd2	Bayesian tobit right-censoring demo
tsls_ d	two-stage least-squares demo
waldf_ d	demo of using wald F-test function

A.1.3 Support functions

program	description
ar1_ like	used by olsar1 (likelihood)
bmapost	used by bma_ g
box_ lik	used by box_ cox (likelihood)
box_ lik2	used by box_ cox2 (likelihood)
chis_ prb	computes chi-squared probabilities
dmult	used by mlogit
egarchm_ lik	likelihood used by egarchm
garchs_ llf	likelihood used by garchs
herg_ llf	likelihood used by emhergott
herg_ llf2	likelihood used by emhergott
hmarkov_ llf	likelihood used by hmarkov_ em
hmarkov_ llf2	likelihood used by hmarkov_ em
fdis_ prb	computes F-statistic probabilities
find_ new	used by bma_ g
grun.dat	Grunfeld's data used by sur_ d
grun.doc	documents Grunfeld's data set
hessian	used by tobit to determine numerical hessian
lo_ like	used by logit (likelihood)
mcov	used by hwhite
mderivs	used by mlogit
mlogit_ lik	used by mlogit
nmlt_ rnd	used by probit_ g
nmrt_ rnd	used by probit_ g, tobit_ g
norm_ cdf	used by probit, pr_ like
norm_ pdf	used by prt_ reg, probit
olse	ols returning only residuals (used by sur)
plt_ eqs	plots equation systems
plt_ reg	plots regressions
pr_ like	used by probit (likelihood)
prt_ eqs	prints equation systems
prt_ gibbs	prints Gibbs sampling models
prt_ reg	prints regressions
prt_ swm	prints switching regression results
sample	used by bma_ g
stdn_ cdf	used by norm_ cdf
stdn_ pdf	used by norm_ pdf
tdis_ prb	computes t-statistic probabilities
to_ like	used by tobit (likelihood)

A.2 Utilities

The utility functions are in a subdirectory `util`.

A.2.1 Utility Function Library

program	description
accumulate	accumulates column elements of a matrix
blockdiag	creates a block diagonal matrix
cal	associates obs # with time-series calendar
ccorr1	correlation scaling to normal column length
ccorr2	correlation scaling to unit column length
cols	returns the # of columns in a matrix or vector
crlag	circular lag function
cumprodc	returns cumulative product of each column of a matrix
cumsumc	returns cumulative sum of each column of a matrix
delif	select matrix values for which a condition is false
diagrv	replaces main diagonal of square matrix with vector
findnear	finds matrix element nearest a scalar value
fturns	finds turning-points in a time-series
growthr	converts time-series matrix to growth rates
ical	associates time-series dates with obs #
indexcat	extract indices equal to a scalar or an interval
indicator	converts a matrix to indicator variables
invccorr	inverse for ccorr1, ccorr2
invpd	makes a matrix positive-definite, then inverts
kernel_ n	normal kernel density estimates
lag	generates a lagged variable vector or matrix
levels	generates factor levels variable
lprint	prints a matrix in LaTeX table-formatted form
lprintf	enhanced lprint function
matadd	adds non-conforming matrices, row or col compatible.
matdiv	divides non-conforming matrices, row or col compatible.
matmul	multiplies non-conforming matrices, row or col compatible.
matsub	divides non-conforming matrices, row or col compatible.
mlag	generates a var-type matrix of lags
mode	calculates the mode of a distribution
mprint	prints a matrix
mprint3	prints coefficient, t-statistics matrices

Utility Function Library - continued

program	description
mth2qtr	converts monthly to quarterly data
nclag	generates a matrix of non-contiguous lags
plt	wrapper function, plots all result structures
prodc	returns product of each column of a matrix
prt	wrapper function, prints all result structures
recserar	recursive AR series (like Gauss)
recsercp	recursive series product (like Gauss)
roundoff	rounds matrix to fixed number of decimal digits
rows	returns the # of rows in a matrix or vector
sacf	sample autocorrelation function estimates
sdiff	seasonal differencing
sdummy	generates seasonal dummy variables
selif	select matrix values for which a condition is true
seqa	a sequence of numbers with a beginning and increment
shist	plots spline smoothed histogram
spacf	sample partial autocorrelation estimates
stdc	std deviations of columns returned as a column vector
sumc	returns sum of each column
tally	computes frequencies of distinct levels
tdiff	time-series differencing
trimc	trims columns of a matrix (or vector) like Gauss
trimr	trims rows of a matrix (or vector) like Gauss
tsdates	time-series dates function
tsprint	print time-series matrix
unsort	unsorts a sorted vector or matrix
vec	turns a matrix into a stacked vector
vech	matrix from lower triangular columns of a matrix
xdiagonal	spreads x($n \times k$) out to X($n * n \times n * k$) diagonal matrix
yvector	repeats y($n \times 1$) to form Y($n * n \times 1$)

A.2.2 demonstration programs

program	description
cal_ d	demonstrates cal function
fturns_ d	demonstrates fturns and plt
ical_ d	demonstrates ical function
lprint_ d	demonstrates lprint function
lprintf_ d	demonstrates lprintf function
mprint_ d	demonstrates mprint function
mprint3_ d	demonstrates mprint3 function
sacf_ d	demonstrates sacf
spacf_ d	demonstrates spacf
tsdate_ d	demonstrates tsdate function
tsprint_ d	demonstrates tsprint function
util_ d	demonstrated some of the utility functions

A.3 Graphing Function Library

A set of graphing functions are in a subdirectory `graphs`.

A.3.1 graphing programs

program	description
tsplot	time-series graphs
pltdens	density plots
pairs	scatter plot (uses histo)
plt	plots results from all functions

A.3.2 Demonstration Programs

program	description
tsplot_ d	demonstrates tsplot
pltdens_ d	demonstrates pltdens
plt_ turnsd	demonstrates plt_ turns
pairs_ d	demonstrates pairwise scatter
plt_ d	demonstrates plt on results structures

A.3.3 Support Functions

program	description
histo	used by pairs
plt_ turns	used by plt to plot turning points

A.4 Regression Diagnostics Library

A library of routines in the subdirectory `diagn` contain the regression diagnostics functions.

A.4.1 regression diagnostic programs

program	description
arch	ARCH(p) test
bkw	BKW collinearity diagnostics
bpagan	Breusch-Pagan heteroskedasticity test
cusums	Brown,Durbin,Evans cusum squares test
dfbeta	BKW influential observation diagnostics
diagnose	compute diagnostic statistics
plt_ dfb	plots dfbetas
plt_ dff	plots dffits
plt_ cus	plots cusums
recresid	compute recursive residuals
rdiag	graphical residuals diagnostics
studentize	standardisation transformation
unstudentize	reverses studentize transformation
qstat2	Box-Ljung Q-statistic

A.4.2 Demonstration Programs

program	description
arch_ d	demonstrates arch
bkw_ d	demonstrates bkw
bpagan_ d	demonstrates bpagan
cusums_ d	demonstrates cusums
dfbeta_ d	demonstrates dfbeta, plt_ dfb, plt_ dff
diagnose_ d	demonstrates diagnose
recresid_ d	demonstrates recresid
rdiag_ d	demonstrates rdiag
unstudentize_ d	demonstrates studentize, unstudentize
qstat2_ d	demonstrates qstat2

A.4.3 support functions

program	description
plt_ cus	plots cusums test results
plt_ dff	plots dffits
../util/plt	plots everything
../regress/ols.m	least-squares regression

A.5 vector autoregressive function library

The vector autoregressive library is in a subdirectory `var_ bvar`.

A.5.1 VAR/BVAR functions

program	description
becm_ g	Gibbs sampling BECM estimates
becmf	Bayesian ECM model forecasts
becmf_ g	Gibbs sampling BECM forecasts
bvar	BVAR model
bvar_ g	Gibbs sampling BVAR estimates
bvarf	BVAR model forecasts
bvarf_ g	Gibbs sampling BVAR forecasts
ecm	ECM (error correction) model estimates
ecmf	ECM model forecasts
irf	impulse response functions
lrratio	likelihood ratio tests for lag length
recm	ecm version of rvar
recm_ g	Gibbs sampling random-walk averaging estimates
recmf	random-walk averaging ECM forecasts
recmf_ g	Gibbs sampling random-walk averaging forecasts
rvar	Bayesian random-walk averaging prior model
rvar_ g	Gibbs sampling RVAR estimates
rvarf	Bayesian RVAR model forecasts
rvarf_ g	Gibbs sampling RVAR forecasts
var	VAR model
varf	VAR model forecasts

A.5.2 Demonstration Programs

program	description
becm_ d -	BECM model demonstration
becm_ g	Gibbs sampling BECM estimates demo
becmf_ d	becmf demonstration
becmf_ gd	Gibbs sampling BECM forecast demo
bvar_ d	BVAR model demonstration
bvar_ gd	Gibbs sampling BVAR demonstration
bvarf_ d	bvarf demonstration
bvarf_ gd	Gibbs sampling BVAR forecasts demo
ecm_ d	ECM model demonstration

A.5.3 Demonstration Programs - continued

program	description
ecmf_ d	ecmf demonstration
irf_ d	impulse response function demo
irf_ d2	another irf demo
lrratio_ d	demonstrates lrratio
pftest_ d	demo of pftest function
recm_ d	RECM model demonstration
recm_ gd	Gibbs sampling RECM model demo
recmf_ d	recmf demonstration
recmf_ gd	Gibbs sampling RECM forecast demo
rvar_ d	RVAR model demonstration
rvar_ gd	Gibbs sampling rvar model demo
rvarf_ d	rvarf demonstration
rvarf_ gd	Gibbs sampling rvar forecast demo
var_ d	VAR model demonstration
varf_ d	varf demonstration

A.5.4 Support Functions

program	description
johansen	used by ecm,ecmf,becm,becmf,recm,recmf
lag	does ordinary lags
mlag	does var-type lags
nclag	does contiguous lags (used by rvar,rvarf,recm,recmf)
ols	used for VAR estimation
pftest	prints Granger F-tests
pgranger	prints Granger causality probabilities
prt	prints results from all functions
prt_ coint	used by prt_ var for ecm,becm,recm
prt_ var	prints results of all var/bvar models
prt_ varg	prints results of all Gibbs var/bvar models
rvarb	used for RVARF forecasts
scstd	does univariate AR for BVAR
theil_ g	used for Gibbs sampling estimates and forecasts
theilbf	used for BVAR forecasts
theilbv	used for BVAR estimation
trimr	used by VARF,BVARF, johansen (in /util/trimr.m)
vare	used by lrratio (vare uses /regress/olse.m)

A.6 Co-integration Library

The co-integration library functions are in a subdirectory `coint`.

A.6.1 Co-integration testing routines

program	description
johansen	carries out Johansen's co-integration tests
adf	carries out Augmented Dickey-Fuller unit root tests
cadf	carries out ADF tests for co-integration
phillips	carries out Phillips-Peron co-integration tests
prt_ coint	prints results from adf,cadf,johansen

A.6.2 Demonstration Programs

program	description
johansen_ d	demonstrates johansen
adf_ d	demonstrates adf
cadf_ d	demonstrates cadf
phillips_ d	demonstrates phillips

A.6.3 Support functions

program	description
c_ sja	returns critical values for SJ maximal eigenvalue test
c_ sjt	returns critical values for SJ trace test
ztcrit	returns critical values for adf test
rztcrit	returns critical values for cadf test
detrend	used by johansen to detrend data series
ptrend	used by adf to create time polynomials
trimr	/util/trimr.m (like Gauss trimr)
cols	/util/cols.m (like Gauss cols)
rows	/util/rows.m (like Gauss rows)
tdiff	/util/tdiff.m differences

A.7 Gibbs sampling convergence diagnostics functions

The Gibbs convergence diagnostic functions are in a subdirectory `gibbs`.

A.7.1 Convergence testing functions

program	description
apm	Geweke's chi-squared test
coda	convergence diagnostics
momentg	Geweke's NSE, RNE
raftery	Raftery and Lewis program Gibbsit for convergence

A.7.2 Demonstration Programs

program	description
apm_ d	demonstrates apm
coda_ d	demonstrates coda
momentg_ d	demonstrates momentg
raftery_ d	demonstrates raftery

A.7.3 Support Functions

program	description
prt_ coda	prints coda, raftery, momentg, apm output (use prt)
empquant	These were converted from:
indtest	Rafferty and Lewis FORTRAN program.
mcest	These function names follow the FORTRAN subroutines
mctest	
ppnd	
thin	

A.8 Distribution functions library

Distribution functions are in the subdirectory `distrib`.

A.8.1 pdf, cdf, inverse functions

program	description
beta_ cdf	beta(a,b) cdf
beta_ inv	beta inverse (quantile)
beta_ pdf	beta(a,b) pdf
bino_ cdf	binomial(n,p) cdf
bino_ inv	binomial inverse (quantile)
bino_ pdf	binomial pdf
chis_ cdf	chisquared(a,b) cdf
chis_ inv	chi-inverse (quantile)
chis_ pdf	chisquared(a,b) pdf
chis_ prb	probability for chi-squared statistics
fdis_ cdf	F(a,b) cdf
fdis_ inv	F inverse (quantile)
fdis_ pdf	F(a,b) pdf
fdis_ prb	probabililty for F-statistics
gamm_ cdf	gamma(a,b) cdf
gamm_ inv	gamma inverse (quantile)
gamm_ pdf	gamma(a,b) pdf
hypg_ cdf	hypergeometric cdf
hypg_ inv	hypergeometric inverse
hypg_ pdf	hypergeometric pdf
logn_ cdf	lognormal(m,v) cdf
logn_ inv	lognormal inverse (quantile)
logn_ pdf	lognormal(m,v) pdf
logt_ cdf	logistic cdf
logt_ inv	logistic inverse (quantile)
logt_ pdf	logistic pdf
norm_ cdf	normal(mean,var) cdf
norm_ inv	normal inverse (quantile)
norm_ pdf	normal(mean,var) pdf
pois_ cdf	poisson cdf
pois_ inv	poisson inverse
pois_ pdf	poisson pdf

pdf, cdf, inverse functions - continued)

program	description
stdn_ cdf	std normal cdf
stdn_ inv	std normal inverse
stdn_ pdf	std normal pdf
tdis_ cdf	student t-distribution cdf
tdis_ inv	student t inverse (quantile)
tdis_ pdf	student t-distribution pdf
tdis_ prb	probabililty for t-statistics

A.8.2 Random Samples

program	description
beta_ rnd	random beta(a,b) draws
bino_ rnd	random binomial draws
chis_ rnd	random chi-squared(n) draws
fdis_ rnd	random F(a,b) draws
gamm_ rnd	random gamma(a,b) draws
hypg_ rnd	random hypergeometric draws
logn_ rnd	random log-normal draws
logt_ rnd	random logistic draws
nmlt_ rnd	left-truncated normal draw
nmrt_ rnd	right-truncated normal draw
norm_ crnd	contaminated normal random draws
norm_ rnd	multivariate normal draws
pois_ rnd	poisson random draws
tdis_ rnd	random student t-distribution draws
unif_ rnd	random uniform draws (lr,rt) interval
wish_ rnd	random Wishart draws

A.8.3 Demonstration and Test programs

program	description
beta_ d	demo of beta distribution functions
bino_ d	demo of binomial distribution functions
chis_ d	demo of chi-squared distribution functions
fdis_ d	demo of F-distribution functions
gamm_ d	demo of gamma distribution functions
hypg_ d	demo of hypergeometric distribution functions
logn_ d	demo of lognormal distribution functions
logt_ d	demo of logistic distribution functions
pois_ d	demo of poisson distribution functions
stdn_ d	demo of std normal distribution functions
tdis_ d	demo of student-t distribution functions
trunc_ d	demo of truncated normal distribution function
unif_ d	demo of uniform random distribution function

A.8.4 Support Functions

program	description
betacfj	used by fdis_ prb
betai	used by fdis_ prb
bincoef	binomial coefficients
com_ size	test and converts to common size
gammalnj	used by fdis_ prb
is_ scalar	test for scalar argument

A.9 Optimisation functions library

Optimisation functions are in the subdirectory `optimiz0e`.

A.9.1 Optimization Functions

program	description
dfp_ min	Davidson-Fletcher-Powell
frpr_ min	Fletcher-Reeves-Polak-Ribiere
maxlik	general all-purpose optimisation routine
pow_ min	Powell conjugate gradient
solvopt	yet another general purpose optimization routine

A.9.2 Demonstration Programs

program	description
optim1_ d	dfp, frpr, pow, maxlik demo
optim2_ d	solvopt demo
optim3_ d	fmins demo

A.9.3 Support Functions

program	description
apprgrdn	computes gradient for solvopt
box_ like1	used by optim3_ d
gradt	computes gradient
hessian	evaluates hessian
linmin	line minimization routine (used by dfp, frpr, pow)
stepsize	stepsize determination
tol_ like1	used by optim1_ d, optim2_ d
updateh	updates hessian

A.10 Spatial Econometrics

A library of spatial econometrics functions is in the subdirectory `spatial`.

A.10.1 Functions

program	description
casetti	Casetti's spatial expansion model
darp	Casetti's darp model
far	1st order spatial AR model - $y = pWy + e$
far_ g	Gibbs sampling Bayesian far model
gwr	geographically weighted regression
bgwr	Bayesian geographically weighted regression
lmerror	LM error statistic for regression model
lmsar	LM error statistic for sar model
lratios	Likelihood ratio statistic for regression models
moran	Moran's I-statistic
sac	spatial model - $y = p * W1 * y + X * b + u, u = c * W2 * u + e$
sac_ g	Gibbs sampling Bayesian sac model
sar	spatial autoregressive model - $y = p * W * y + X * b + e$
sar_ g	Gibbs sampling Bayesian sar model
sarp_ g	Gibbs sampling Bayesian sar Probit model
sart_ g	Gibbs sampling Bayesian sar Tobit model
sem	spatial error model - $y = X * b + u, u = c * W + e$
sem_ g	Gibbs sampling Bayesian spatial error model

Functions - continued

program	description
semo	spatial error model (optimization solution)
sdm	spatial Durbin model $y = a + X * b1 + W * X * b2 + e$
sdm_ g	Gibbs sampling Bayesian spatial Durbin model
walds	Wald test for regression models
xy2cont	constructs a contiguity matrix from x-y coordinates

A.10.2 Demonstration Programs

program	description
casetti_ d	Casetti model demo
darp_ d	Casetti darp demo
darp_ d2	darp for all data observations
far_ d	demonstrates far using a small data set
far_ d2	demonstrates far using a large data set
far_ gd	far Gibbs sampling with small data set
far_ gd2	far Gibbs sampling with large data set
gwr_ d	geographically weighted regression demo
gwr_ d2	GWR demo with Harrison-Rubinfeld Boston data
bgwr_ d	demo of Bayesian GWR
bgwr_ d2	BGWR demo with Harrison-Rubinfeld Boston data
lmerror_ d	lmerror demonstration
lmsar_ d	lmsar demonstration
lratios_ d	likelihood ratio demonstration
moran_ d	moran demonstration
sac_ d	sac model demo
sac_ d2	sac model demonstration large data set
sac_ gd	sac Gibbs sampling demo
sac_ gd2	sac Gibbs demo with large data set
sar_ d	sar model demonstration
sar_ d2	sar model demonstration large data set
sar_ gd	sar Gibbs sampling demo
sar_ gd2	sar Gibbs demo with large data set
sarp_ gd	sar Probit Gibbs sampling demo
sart_ gd	sar Tobit model Gibbs sampling demo
sdm_ d	sdm model demonstration
sdm_ d2	sdm model demonstration large data set
sdm_ gd	sdm Gibbs sampling demo
sdm_ gd2	sdm Gibbs demo with large data set
sem_ d	sem model demonstration
sem_ d2	sem model demonstration large data set

Demonstration Programs - continued

program	description
sem_ gd	sem Gibbs sampling demo
sem_ gd2	sem Gibbs demo with large data set
semo_ d	semo function demonstration
semo_ d2	semo demo with large data set
walds_ d	Wald test demonstration
xy2cont_ d	xy2cont demo

A.10.3 Support Functions

program	description
anselin.dat	Anselin (1988) Columbus crime data
boston.dat	Harrison-Rubinfeld Boston data set
latit.dat	latittude for HR data
longi.dat	longitude for HR data
c_ far	used by far_ g
c_ sem	used by sem_ g
c_ sar	used by sar_ g
c_ sdm	used by sdm_ g
c_ sac	used by sac_ g
darp_ lik1	used by darp
darp_ lik2	used by darp
elect.dat	Pace and Barry 3,107 obs data set
ford.dat	Pace and Barry 1st order contiguity matrix
f_ far	far model likelihood (concentrated)
f_ sac	sac model likelihood (concentrated)
f_ sar	sar model likelihood (concentrated)
f_ sem	sem model likelihood (concentrated)
f_ sdm	sdm model likelihood (concentrated)
f2_ far	far model likelihood
f2_ sac	sac model likelihood
f2_ sar	sar model likelihood
f2_ sem	sem model likelihood
f3_ sem	semo model likelihood
f2_ sdm	sdm model likelihood
normxy	isotropic normalization of x-y coordinates
prt_ gwr	prints gwr_ reg results structure
prt_ spat	prints results from spatial models
scoref	used by gwr
wmat.dat	Anselin (1988) 1st order contiguity matrix