Computer Tutorial 1: Matlab Basics

Most Bayesians econometricians create their own programs and Matlab is probably the most popular language for doing so. When writing my textbooks and monograph I used Matlab and the websites associated with them contain many sample programs. Matlab is a powerful program that is used in a variety of scientific disciplines. In this course, I will not have time to teach you everything about Matlab. Rather I attempt to teach you the basics necessary to for you to carry out Bayesian inference in the models considered in this course. With a knowledge of these basics, you can then use Matlab online help or manuals to figure out more complicated things. My way of teaching the basics of Matlab will be indirect: I will provide sample programs and then get you to figure out what each step does. Then I will ask you to create similar programs. If you do not like this style of teaching and prefer a more systematic tutorial about Matlab, there are many web-based resources you should feel free to use. Wikiuniversity has a free online course: https://en.wikiversity.org/wiki/MATLAB_essential. Matlab itself has online tutorials which you may want to experiment with. Many lecturers have put together tutorial materials that you may find useful (e.g. http://www.tcd.ie/Economics/TEP/2010/TEP0110.pdf).
For questions 4 through 6 Matlab code is available on the course website. For the first three exercises, I want you to program things up yourself.

Overview

Matlab is a matrix programming language. Unlike other programs you may have used (e.g. Microfit, Eviews, Stata, etc.), Matlab is exclusively a programming language. For instance, in these other languages you would just click on a button to do OLS or type a simple command (e.g. "regress y x"). In Matlab you have to actually program up OLS. The OLS estimator is given as:

$$\widehat{\beta} = (X'X)^{-1} X'y.$$

In Matlab, the format for this command is:
bhat = inv(x'*x)*x'y;
and you would actually have to type this line and then run the program containing it.
A Matlab program is a series of commands such as this. These commands will be listed in a file called an m-file (with extension ".m" so a program name might be myprogram.m). Matlab has a text editor which allows you to create an m-file (click on "File/New"). Once you have created and saved such a file using the Matlab editor, you can then run the program. You can do this by clicking on the "run" icon which has a green triangle in it (located at the top of the text editor). Matlab has many different windows, but two of them are the most important: the Command window (where you can type commands and where the results of the program are displayed) and the text editor (where you write and run your code). I will demonstrate these general features of Matlab in the computer lab. Below I will focus on the program writing.
Lines which begin with % are ignored by Matlab. Programmers often place comments in their programs in this way to explain key steps in their programs. Note also that each Matlab command ends in a ";" (although there are some exceptions to this such as for/end/if statements which do not require the ";" although it is harmless if you add it).

Exercise 1: *Basic Matrix Commands*
A sample program

```
%This is a simple program which illustrates Matlab matrix commands
x = [1 2 3; 4 5 6; 7 8 9];
disp x;
disp(x);
y = [11 12 13; 14 15 16; 17 18 19];
disp y;
disp(y);
z=x+y;
disp z ;
disp(z);
w=x-y;
```

```
disp w;
disp(w);
u= x*y;
disp u;
disp(u);
a=[x, y];
disp a;
disp(a);
b=[x; y];
disp b;
disp(b);
c=x(:,2);
disp c;
disp(c);
d=y(:,1);
disp d;
disp(d);
e=x(2,3);
disp e
disp(e);
```

a) Create the preceding program and run it in Matlab and examine the output Matlab produces. Describe what each line of this program does.
b) Add a line to this program which creates a new matrix, f, which is the transpose of x.
c) Add a line to this program which creates a new matrix, g, which is the identity matrix plus x.
d) Add a line to this program which creates a new matrix, ginv, which is the inverse of g. What happens if you try to take the inverse of x?

Exercise 2: *OLS Estimation Using Artificial Data*

Matlab has many scripts (i.e. built-in little programs) that you can call automatically as part of your program. (Note: You can also create your own scripts). Here we show you how to use Matlab's scripts for random number generation from the Normal and Uniform distributions to create an artificial data set from the regression model: $y_i = \alpha + \beta x_i + \varepsilon_i$ for $i = 1, .., 100$. We set $\alpha = 1$, $\beta = 2$ and let the x_i and ε_i be random draws from the $U(0,1)$ and $N(0,1)$ distribution, respectively.

A sample program

```
%This is a program which artificially creates a data set and then does OLS estimation using it
%First part of this program artificially simulates data
n=100;
alpha=1;
beta=2;
e = randn(n,1);
x=rand(n,1);
y=alpha + x*beta + e;
%following line adds intercept to x. explain why
x=[ones(n,1), x];
%Following part of the program does OLS estimation
bhat = inv(x'*x)*x'*y;
disp 'The OLS estimate of beta is';
disp(bhat);
%the OLS residuals
resids = y - x*bhat;
%The OLS estimate of the error variance
s2 = resids'*resids/(n-2);
disp 'The OLS estimator of the error variance is';
disp(s2);
```

a) Create this program and run it in Matlab and examine the output in Matlab. Describe what each line of this program does.
b) Extend this program to calculate the R^2 of this regression and print out the result.
c) Extend this program to calculate the covariance matrix of the OLS estimators (i.e. $var\left(\widehat{\beta}_{OLS}\right) = s^2\left(X'X\right)^{-1}$) and print out the result.

Exercise 3: *For Loops and If Statements*

When doing Monte Carlo integration or Gibbs sampling we repeatedly take draws from the posterior distribution. Matlab does this kind of repeated action using constructs called "for loops".

A sample program

```
%This is a program which illustrates for loops and if statements
%first create a column vector to work with
x=[1;2;7;5;9;3;6;9;1;11;1];
%the following command sums up the elements of a column vector
xsum=sum(x);
xsum1=0;
for i=1:11
xsum1=xsum1 + x(i,1);
end
disp xsum;
disp(xsum);
disp xsum1;
disp(xsum1);
%now illustrate the if command
xsum2=0;
for i=1:11
if x(i,1)>4
xsum2=xsum2 + x(i,1);
end
end
disp xsum2
disp(xsum2);
```

a) Create this program and run it in Matlab and examine the output. Describe what each line of this program does. In particular, why are xsum1 and xsum the same as one another? What does the "if" statement do? What is "xsum2"?
b) The sample program *sums* various column vectors. Modify this program to calculate *averages* (i.e. means).

Exercise 4: *Drawing from Standard Distributions*

Simulation-based inference using algorithms such as the Gibbs sampler requires the researcher to be able to draw from standard distributions. In this exercise we discuss how MATLAB can be used to obtain draws from a variety of standard continuous distributions. Specifically, we obtain draws from the Uniform, Normal, Student-t, Beta, Exponential and Chi-squared distributions (see the Appendix of Bayesian Econometric Methods for definitions of these distributions). Using the Matlab program for this exercise (Ex4.m), obtain sets of 10, 100 and 100,000 draws from the Uniform, standard Normal, Student-t(3) (denoted $t(0,1,3)$ in the notation of the Appendix to the book), Beta(3,2), Exponential with mean 5 and $\chi^2(3)$ distributions. For each sample size calculate the mean and standard deviation and compare these quantities to the known means and standard deviations from each distribution.

Exercise 5: *Monte Carlo Integration*

If the posterior density $p(\theta|y)$ takes the a familiar form (e.g. a Normal or Student-t or Gamma or other distribution for which computer algorithms exist to take random draws) then we can obtain R i.i.d. draws of the parameters, which we denote $\theta^{(r)}$, $r = 1, ..., R$. Usually, quantities of interest to the researcher are

functions of the model parameters. Let us call such a function $g(\theta)$. The researcher would then often be interested in calculating:

$$E(g(\theta)|y) = \int g(\theta)\, p(\theta|y)\, d\theta$$

Monte Carlo integration allows us to calculate integrals of this form. The weak law of large numbers implies that

$$E(g(\theta)|y) \simeq \frac{\sum_{r=1}^{R} g\left(\theta^{(r)}\right)}{R}$$

This means that the posterior mean of $g(\theta)$ can be calculated by drawing from the posterior and then averaging functions of the posterior draws. Exercise: Suppose $p(\theta|y) \sim N(1,4)$ and the quantity of interest is $g(\theta) = \theta^2$. Use Monte Carlo integration to calculate $E\left(\theta^2\right)$. Code for this question is in Ex5.m. Note: in this case, you know the correct answer is $E\left(\theta^2\right) = 5$ (since the definition of variance tells you that $var(\theta) = E\left(\theta^2\right) - [E(\theta)]^2$ and, in this exercise, $var(\theta) = 4$ and $E(\theta) = 1$), so you would not need to have done Monte Carlo integration. Optional exercise: modify Ex5.m to calculate the posterior mean for a more complicated quantity of interest for which analytical results are not so easily available (e.g. calculate $\Pr\left(\theta^2 > 2\right)$ or $E(\ln(\theta))$ or some other choice for $g(\theta)$).

Exercise 6: *Gibbs Sampling from the Bivariate Normal*
The purpose of this question is to learn about the properties of the Gibbs sampler in a very simple case. Assume that you have a model which yields a bivariate Normal posterior,

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right),$$

where $|\rho| < 1$ is the (known) posterior correlation between θ_1 and θ_2.
(a)Write a program which uses Monte Carlo integration to calculate the posterior means and standard deviations of θ_1 and θ_2.
(b) Write a program which uses Gibbs sampling to calculate the posterior means and standard deviations of θ_1 and θ_2.
(c) Set $\rho = 0$ and compare the programs from parts a) and b) for a given number of replications (e.g. $R = 100$) and compare the accuracy of the two algorithms.
(d) Repeat part (c) of this question for $\rho = .5, .9, .99$ and $.999$. Discuss how the degree of correlation between θ_1 and θ_2 affects the performance of the Gibbs sampler. Make graphs of the Monte Carlo and Gibbs sampler replications of θ_1 (i.e. make a graph with x-axis being replication number and y-axis being θ_1). What can the graphs you have made tell you about the properties of Monte Carlo and Gibbs sampling algorithms?
(e) Repeat parts (c) and (d) using more replications (e.g. $R = 50,000$) and discuss how Gibbs sampling accuracy improves with number of replications.