Matlab Handout # 4

GMM Estimation

This practical discusses a sequence of functions that are used to implement GMM estimation and calculate related statistics of interest. In addition, we will demonstrate the use of these functions by estimating the Hansen and Singleton (1982) version of the consumption based asset pricing model. All the files used for these demonstration must be downloaded from i:\Ecg790ah to your k drive.

(i) Overview

The estimation is implemented using a set of functions that you can find in the GMM toolbox. To access a list of these functions, along with a short description, enter the following in Matlab’s command window

```matlab
>> help gmmtbx
```

You should see a list of all the functions used for the GMM estimation, with a short description alongside. If you need help with a specific function, for example, `gmmest` that controls the main GMM estimation, you can enter the following

```matlab
>> help gmmest
```

In this case, you will get a list explaining the syntax, input, and output arguments of the function. These are described in the following section.

(ii) Application specific inputs

Data

The first step is to create a variable that contains the dataset used for the estimation. You are free to create this variable in any possible way, and give to it any name you like. For illustration purposes, suppose that the name you give is `dataset`.

Starting parameters

The second step is to create some `starting parameters` for the GMM estimation. Specifically, you need to create:

- A `first step weighting matrix`. This can be the identity matrix, the inner product of the instruments, or any other matrix you like. Suppose the name you give to the weighting
matrix is \textit{step1W}.

- A set of \textit{initial GMM estimates}. The GMM estimation is an iterative procedure; as such, you need a set of starting values for the parameter vector. The only requirement for the starting values is that the user supplies them in a column-vector form. For example, if you have to estimate 6 parameters, the only requirement is that the starting values must be in a $6 \times 1$ vector. Suppose the name you give is \textit{initialtheta}.

\textbf{Functions}

This can be considered as the most crucial part of your program. The user must create a function that calculates the moments and their gradient. The name of the function can be anything you like. Concerning the syntax of the function, it must be written in the following way

\[ \text{[mom, gradmom]} = \text{name(data, parameters, whatever)} \]

This function returns two output arguments: \textit{mom} is the matrix of the moments. For a model with $T$ observations and $m$ moment conditions, the \textit{mom} is a $T \times m$ matrix. \textit{gradmom} is a matrix with the first derivative of the moments. For a model with $m$ moments and $p$ parameters, this must be an $m \times p$ matrix.

The inputs you supply are the dataset (\textit{data}), the vector of the parameters at which you evaluate the function (\textit{parameters}), and \textit{whatever} is any other additional argument that your function needs; the most usual additional argument is the instruments, in the case of GIV estimation.

\textit{Important:} The dimensions of \textit{mom}, and \textit{gradmom} must be the one stated above; otherwise, the estimation method will not work. Finally, note that the name for the outputs, inputs, and the function can be whatever you like. For example, in the case of the Hansen CBAPM model, you may use the following syntax

\[ [\text{moments, gradmoments}] = \text{cbapm(data, parameters, instruments)} \]

\textbf{Setable options}

There are some options that you need to set in order to do GMM estimation. These are described below:

- \textbf{center} A dummy variable, taking the values 0 or 1.
  - If you set it to 0, the variance of the moments is calculated using the uncentered moment conditions.
  - If you set it to 1, the variance of the moments is calculated using the centered moment conditions.

- \textbf{method} Covariance matrix estimation method. Set this option equal to
  - \textit{’HAC_B’}, for HAC with Bartlett kernel.
  - \textit{’HAC_P’}, for HAC with Parzen kernel.
  - \textit{’SerUnc’}, if the moments are martingale differences.
bandw  The desired bandwidth, used in HAC estimation. The bandwidth must be a non-negative integer.
  ##If the bandwidth is not given, and a HAC estimator has been selected by the user, the "optimal" bandwidth is automatically calculated using Newey and West’s Method of Bandwidth Selection

itergmm Maximum number for the iterated GMM estimator.
  ##(The default is 100).

tol  Tolerance criterion for the iteration procedure.
  ##(The default is 1e-006).

maxfeval Maximum number of function evaluations (This option is used by fminunc).
  ##(The default is 100*p, where p is the number of parameters).

miter Maximum number of function iterations (this option is used by fminunc).
  ##(The default is 100).

All these options can be set using the function optset. Specifically, any of these options can be set using the following command:
  >>optset('gmmest', 'option name', option value);
For example, if you want to use the Bartlett kernel with a bandwidth of 8, you type the following commands:
  >>optset('gmmest', 'method', 'HAC_B');
  >>optset('gmmest', 'bandw', 8);

Using the gmmest function
The last step is to use the GMM estimation command. After following all the above steps, the command you have to use is:
  >>[thetafinal, Jtest, probJ, Sfinal, finalmom, finalmomgrad, vartheta, stdtheta, ci] = gmmest(data,
       'popmom', stval, WM, whatever);

Inputs  In the above function, the inputs, in order of appearance, are the dataset, the name of the function that calculates the moments and their gradient, a vector with the starting parameter values, the first step weighting matrix, and "whatever" extra argument that you need for the estimation. The most common extra argument is the matrix of the instruments.

Output  The output arguments, in the order that the variables appear, are the final GMM estimates, the J test of overidentifying conditions, the probability value of the J test, the covariance matrix of the moments, the value of the moments, the gradient of the moments, and the variance, standard deviation and confidence interval of the GMM estimates.
For example, using the variable names we set earlier, the gmmest function can be used in the following way:

```matlab
>> [theta, J, pJ, Sfinal, mom, momgrad, vartheta, stdtheta, ci] = gmmest(dataset, 'cbapm', initialtheta, step1W, instruments);
```

Note that the "output" variables can have any name you like.

(iii) Example for the moments function

Suppose that you want to estimate a model for which the moment conditions are:

\[
E[\theta_1 X_{1,t} - \theta_2 X_{2,t}]Z_t = 0 \\
E[\theta_3 X_{3,t} - \theta_2 \theta_3 (X_{1,t}X_{3,t})]Z_t = 0
\]

where \( \theta_i, i = 1, 2, 3 \), are the parameters of interest, \( X_i, i = 1, 2, 3 \), are observed, non stochastic, variables, and \( Z_t \) is an \( 1 \times 4 \) vector of instruments.

In addition, suppose that you have a \( T \times 3 \) matrix \( X \), containing \( T \) observations on the \( X_i \)'s and a \( T \times 4 \) matrix \( Z \), containing \( T \) observations on the instruments. Your objective is to estimate the parameters using GMM, and you need to create a function that calculates the moments and their gradient. Note that you have 8 moment conditions and 3 parameters; as a result, the gradient must be an \( 8 \times 3 \) matrix. Taking the first derivatives of the above conditions with respect to \( \theta_i, i = 1, 2, 3 \), gives:

\[
\begin{bmatrix}
E[X_{1,t}Z_t] & E[-X_{2,t}Z_t] & 0_{4 \times 1} \\
0_{1 \times 4} & E[-\theta_3 \theta_2 \theta_3^{-1} (X_{1,t}X_{3,t})Z_t] & E[X_{3,t} - ln(\theta_2) \theta_2 \theta_3 (X_{1,t}X_{3,t})]Z_t
\end{bmatrix}
\]

Having define the moments and their gradient, we can now proceed to writing the Matlab function that will calculate them numerically.

The following code provides the function we want.

```matlab
% The sample code starts here
function [mom, grad] = example(data, theta, instruments);

% Define the parameters of interest
theta1 = theta(1,1);
theta2 = theta(2,1);
theta3 = theta(3,1);

% Extract the three different X's from the data matrix
x1 = data(:,1);
x2 = data(:,2);
x3 = data(:,3);

% Extract the four different instruments from the instrument matrix
Z1 = instruments(:,1);
Z2 = instruments(:,2);
```
Z3 = instruments(:,3);
Z4 = instruments(:,4);

% Get the size of the dataset
T = size(data,2);

% Calculate the moments
m1 = (theta1*x1 - theta2*x2).*Z1;
m2 = (theta1*x1 - theta2*x2).*Z2;
m3 = (theta1*x1 - theta2*x2).*Z3;
m4 = (theta1*x1 - theta2*x2).*Z4;
m5 = (theta3*x2 - (theta2*theta3)*x2.*x1).*Z1;
m6 = (theta3*x2 - (theta2*theta3)*x2.*x1).*Z2;
m7 = (theta3*x2 - (theta2*theta3)*x2.*x1).*Z3;
m8 = (theta3*x2 - (theta2*theta3)*x2.*x1).*Z4;
mom = [m1 m2 m3 m4 m5 m6 m7 m8];

% Calculate the gradient of the moments
g11 = sum(x1.*Z1)/T;
g21 = sum(x1.*Z2)/T;
g31 = sum(x1.*Z3)/T;
g41 = sum(x1.*Z4)/T;
g12 = -sum(x2.*Z1)/T;
g22 = -sum(x2.*Z2)/T;
g32 = -sum(x2.*Z3)/T;
g42 = -sum(x2.*Z4)/T;
g52 = -theta2*(theta3-1)*sum(x1.*x3.*Z1)/T;
g62 = -theta2*(theta3-1)*sum(x1.*x3.*Z2)/T;
g72 = -theta2*(theta3-1)*sum(x1.*x3.*Z3)/T;
g82 = -theta2*(theta3-1)*sum(x1.*x3.*Z4)/T;
g53 = sum((x2-log(theta3))*theta2*(theta3)*x1.*x3).*Z1)/T;
g63 = sum((x2-log(theta3))*theta2*(theta3)*x1.*x3).*Z2)/T;
g73 = sum((x2-log(theta3))*theta2*(theta3)*x1.*x3).*Z3)/T;
g83 = sum((x2-log(theta3))*theta2*(theta3)*x1.*x3).*Z4)/T;
grd = [g11 g21 0;g21 g22 0;g31 g32 0;g41 g42 0;g52 g53; 0g62 g63;0 g72 g73;0 g82 g83];

% The sample code finishes here

Note that in the above code, mom is a $T \times 8$ matrix, and grad is an $8 \times 3$ matrix. Finally, note that the above is just a sample code; you can create the mom and grad function in other ways as well (e.g. using "for" loops, etc.).