Econometrics - Problem Set 4

Gualtiero Azzalini

Exercise 1

The following answer assume $H_0 = a(\theta_0) = 0$, $\tilde{Q_n} = -g_n(\theta)' \hat{W} g_n(\theta)$, optimal weighting, i.e. $\hat{W} \xrightarrow{p} S^{-1}$ and that all regularity conditions in Chapter 4 in the notes hold.

Part a

Wald

The usual Wald statistic under $H_0 = a(\theta_0) = 0$ is:

$$\xi_W = na\left(\hat{\theta}\right)' \left(A\left(\hat{\theta}\right)\hat{\Omega}A\left(\hat{\theta}\right)'\right)^{-1} a\left(\hat{\theta}\right).$$

As the statistic does not contain $\widetilde{Q_n}(\theta)$ and the maximization is not affected by the lack of $1/2$, the Wald statistic has in this case the same asymptotic distribution:

$$\xi_W \xrightarrow{d} \chi^2_r.$$

LM

In this case we have:

$$\tilde{\xi}_{LM} = n\left(\frac{\partial\tilde{Q}_n\left(\tilde{\theta}\right)}{\partial\theta}\right)' \hat{V}^{-1}\left(\frac{\partial\tilde{Q}_n\left(\tilde{\theta}\right)}{\partial\theta}\right) = n2\left(\frac{\partial Q_n\left(\tilde{\theta}\right)}{\partial\theta}\right)' \hat{V}^{-1}2\left(\frac{\partial Q_n\left(\tilde{\theta}\right)}{\partial\theta}\right) = 4\xi_{LM}$$

$$\Rightarrow \tilde{\xi}_{LM} \xrightarrow{d} 4\chi^2_r$$

$$\Rightarrow \tilde{\xi}_{LM} \xrightarrow{d} \chi^2_{4r}.$$

QLR

Using the same computations as before, and in particular (1), substituting into (4.46) in the notes the relevant formulas in this case we get:

$$\tilde{\xi}_{QLR} = 2n\left[\tilde{Q}_n\left(\hat{\theta}\right) - \tilde{Q}_n\left(\tilde{\theta}\right)\right] = 2n\left[2Q_n\left(\hat{\theta}\right) - 2Q_n\left(\tilde{\theta}\right)\right] = 2\xi_{QLR}$$

$$\Rightarrow \tilde{\xi}_{QLR} \xrightarrow{d} 2\chi^2_r$$

$$\Rightarrow \tilde{\xi}_{QLR} \xrightarrow{d} \chi^2_{2r}.$$

Part b

As described above, for the Wald test nothing changes as the asymptotic distribution remains χ^2_r. However, for the LM and QLR tests the asymptotic distribution is different and, specifically, it is respectively χ^2_{4r} and χ^2_{2r}. This means:

- That even if we consider the optimally weighted case the three tests are not asymptotically equivalent
- The test statistic is going to be lower, in the sense that $\mathbb{P}\left(\chi^2 < \tilde{\xi}\right)$ decreases as the degrees of freedom increase, which may lead to more frequent non rejection compared to the baseline case (consequently this implies more probability of Type II error). Given the distributions of the LM and QLR above depicted this phenomenon is going to be more relevant in the LM case.

Exercise 2

I answer this question under the 3-instruments case. I report here the first part of the code which is exactly the same I used in the previous problem set to get the two-step GMM and the confidence intervals (which I report because I need them in exercise 3 part a).

```
%% Exercise 2  clear;
% Import the .csv file
M = csvread('hsdata.csv',1,0);
t = csvread('hsdata.csv',1,0,[1,0,250,0])';
c_or = csvread('hsdata.csv',1,1,[1,1,250,1])';
d_or = csvread('hsdata.csv',1,2,[1,2,250,2])';
r_or = csvread('hsdata.csv',1,3,[1,3,250,3])';
data = [c_or;d_or;r_or];
% Balance the observations
for i=1:(length(c_or)-1)
c_1(i)=c_or(i+1);
r_1(i)=r_or(i+1);
d_1(i)=d_or(i+1);
end
c=c_or;d=d_or;r=r_or;
r(250)=[];c(250)=[];d(250)=[];
T = length(c);
z=[ones(1,T); exp(r); (1+c)];

% Compute GMM - first step
```

```
W_1=eye(3);
fun1 = @(x)objective(x, W_1, z, T, c_1, r_1);
x0=[0.9 0.9];
[theta1_gmm, Qval1] = fminsearch(fun1, x0);

% Compute GMM - second step
% Euler conditions at parameters estimated in step 1
eul = ee(theta1_gmm, z, T, c_1, r_1);
% 2nd-step weighting matrix - the long-run variance, without lags (mds)
W_2=inv(longrunW(eul, T, 0));
fun2 = @(x)objective(x, W_2, z, T, c_1, r_1);
[theta2_gmm, Qval2]= fminsearch(fun2, theta1_gmm);

% Compute Asymptotic variance (G'S^(-1)G)^(-1) using consistent estimators
G = [sum(exp(-theta2_gmm(2).*c_1+r_1))/T,
sum(-c_1.*theta2_gmm(1).*exp(-theta2_gmm(2).*c_1+r_1))/T;
sum(exp(-theta2_gmm(2).*c_1+r_1+r))/T,
sum(-c_1.*theta2_gmm(1).*exp(-theta2_gmm(2).*c_1+r_1+r))/T;
sum(exp(-theta2_gmm(2).*c_1+r_1).*(1+c))/T,
sum(-c_1.*theta2_gmm(1).*exp(-theta2_gmm(2).*c_1+r_1).*(1+c))/T];
eul2 = ee(theta2_gmm, z, T, c_1, r_1);
W_2_2=inv(longrunW(eul2, T, 0));
AVar = inv(G'*W_2_2*G);
delta_CI = [-1.96*sqrt(AVar(1,1)/(T))+theta2_gmm(1)
1.96*sqrt(AVar(1,1)/(T))+theta2_gmm(1)];
gamma_CI = [-1.96*sqrt(AVar(2,2)/(T))+theta2_gmm(2)
1.96*sqrt(AVar(2,2)/(T))+theta2_gmm(2)];
```

Part a

I have to test the null hypothesis $H_0: \ \delta = 0.99 \ and \ \gamma = 4.$ The code I used for the three tests follows:

```
% Test1 H0: delta=0.99, gamma=4 %
null_1 = [0.99 4];
A_theta_1 = [1 0;0 1];

% WALD
wald_1 = T*(theta2_gmm-null_1)*inv(A_theta_1*AVar*A_theta_1')*(theta2_gmm-null_1)';
pvalue_wald_1 = 1-chi2cdf(wald_1,2);

% LM %
theta2_cons_1 = null_1;
% Euler conditions at parameters estimated in step 1
eul_c_1 = ee(theta2_cons_1, z, T, c_1, r_1);
```

```
% 2nd-step weighting matrix - the long-run variance, without lags (mds)
W_2_cons_1=inv(longrunW(eul_c_1, T, 0));
Qval_cons2_1 = objective(theta2_cons_1, W_2_cons_1, z, T, c_1, r_1);
% Compute Asymptotic variance (G'S^(-1)G)^(-1) for constrained
G_cons_1 = [sum(exp(-theta2_cons_1(2).*c_1+r_1))/T,
sum(-c_1.*theta2_cons_1(1).*exp(-theta2_cons_1(2).*c_1+r_1))/T;
 sum(exp(-theta2_cons_1(2).*c_1+r_1+r))/T,
sum(-c_1.*theta2_cons_1(1).*exp(-theta2_cons_1(2).*c_1+r_1+r))/T;
  sum(exp(-theta2_cons_1(2).*c_1+r_1).*(1+c))/T,
sum(-c_1.*theta2_cons_1(1).*exp(-theta2_cons_1(2).*c_1+r_1).*(1+c))/T];

AVar_cons_1 = inv(G_cons_1'*W_2_cons_1*G_cons_1);
eul_c_1 = ee(theta2_cons_1, z, T, c_1, r_1);
lm_1 = T.*(-(T^(-1)).*G_cons_1'*W_2_cons_1*transpose(sum(eul_c_1')))'*AVar_cons_1*...
  (-(T^(-1)).*G_cons_1'*W_2_cons_1*transpose(sum(eul_c_1')));
pvalue_lm_1 = 1-chi2cdf(lm_1,2);

% QLR %
qlr_1 = -2*T*(Qval2-Qval_cons2_1);
pvalue_qlr_1 = 1-chi2cdf(qlr_1,2);
```

To perform the Wald test, I define the null (I will test that the difference between θ_0 and the null is zero) and the $A(\hat{\theta})$ matrix of derivatives which in this case is simply a diagonal matrix as restrictions are linear and depend only on the relevant parameter. I use the asymptotic variance of the GMM computed before as consistent estimator of the variance (as we consider the optimally weighted case). For the LM test, instead, I have to compute first the constrained estimator under the restrictions in H_0. To do that I simply replace the two values in the objective function and in the functions I use to compute the variance. Finally for the QLR test I simply recover the max values of the criterion function under the two-step GMM and the constrained estrimators and I multiply by $2T$. Note there is a minus in my formula just because I minimize the inverse of the criterion function. The table below reports the values of the three t-stats ξ_W,ξ_{LM},ξ_{QLR} and their p-values . For all the three cases the asymptotic distribution is a χ^2_2 as there are two restrictions. By looking at the table two things emerge: first, that the tests are asymptotically equivalent as the t-stats and p-values are almost the same and second that they all don't reject the null hypothesis.

	ξ	p-value
Wald	0.31377	0.8548
LM	0.29731	0.86187
QLR	0.31494	0.8543

Part b

I have to test the null hypothesis $H_0: \ \delta = 0.99$. The code I used for the three tests follows:

```
% Test2 H0: delta=0.99 %
null_2 = 0.99;
```

```
A_theta_2 = [1 0]';

% WALD %
wald_2=T*(theta2_gmm*A_theta_2-null_2)*inv(A_theta_2'*AVar*A_theta_2)...
*(theta2_gmm*A_theta_2-null_2)';
pvalue_wald_2 = 1-chi2cdf(wald_2,1);

% LM %
% Compute constrained GMM - first step
W_1=eye(3);
fun3 = @(x)objective([0.99 x], W_1, z, T, c_1, r_1);
x0=[0.9];
gamma1_cons = fminsearch(fun3, x0);
theta1_cons_1=[0.99 gamma1_cons];
% Compute GMM constrained - second step
% Euler conditions at parameters estimated in step 1
eul_c_2 = ee(theta1_cons_1, z, T, c_1, r_1);
% 2nd-step weighting matrix - the long-run variance, without lags (mds)
W_2_cons_2=inv(longrunW(eul_c_2, T, 0));
fun4=@(x)objective([0.99 x], W_2_cons_2, z, T, c_1, r_1);
[gamma2_cons Qval_cons2_2]= fminsearch(fun4, gamma1_cons);
theta2_cons_2=[0.99 gamma2_cons];
% Compute Asymptotic variance (G'S^(-1)G)^(-1) for constrained
G_cons_2 = [sum(exp(-theta2_cons_2(2).*c_1+r_1))/T,
sum(-c_1.*theta2_cons_2(1).*exp(-theta2_cons_2(2).*c_1+r_1))/T;
 sum(exp(-theta2_cons_2(2).*c_1+r_1+r))/T,
sum(-c_1.*theta2_cons_2(1).*exp(-theta2_cons_2(2).*c_1+r_1+r))/T;
  sum(exp(-theta2_cons_2(2).*c_1+r_1).*(1+c))/T,
sum(-c_1.*theta2_cons_2(1).*exp(-theta2_cons_2(2).*c_1+r_1).*(1+c))/T];
eul_c_2 = ee(theta2_cons_2, z, T, c_1, r_1);
W_2_cons_2=inv(longrunW(eul_c_2, T, 0));
AVar_cons_2 = inv(G_cons_2'*W_2_cons_2*G_cons_2);
lm_2 = T.*(-(T^(-1)).*G_cons_2'*W_2_cons_2*transpose(sum(eul_c_2')))'*AVar_cons_2*...
  (-(T^(-1)).*G_cons_2'*W_2_cons_2*transpose(sum(eul_c_2')));
pvalue_lm_2 = 1-chi2cdf(lm_2,1);

% QLR %
qlr_2 = -2*T*(Qval2-Qval_cons2_2);
pvalue_qlr_2 = 1-chi2cdf(qlr_2,1);
```

Now the null hypothesis regards only the parameter δ, hence I have to change the null and the $a\left(\hat{\theta}\right)$ matrix in the Wald test as we are testing only one restriction. In fact, $a\left(\hat{\theta}\right) = [1\ 0]'$ (transposed because my gmm estimator is a row vector) so that $\hat{\theta}_{GMM,2} a\left(\hat{\theta}\right) = \delta$ is a scalar. Again, as the restriction is linear in δ and it does not depend on γ the matrix of derivatives is just the vector [1 0]. To compute the LM and QLR tests I have to calculate the constrained estimator. Note that as there is only one constraint I have to estimate the two-step constrained GMM by taking as given the value of δ given in the null hypothesis and maximize with respect to γ. Hence in the code part of the LM test I compute this GMM constrained estimator by using the usual procedure even though I maximize only with respect to γ. Finally for the QLR test I simply recover the max values of the criterion function under the two-step GMM and the constrained estrimators and I multiply by $2T$ (again there is a minus because I minimize the inverse of the objective). The table below reports the values of the three t-stats ξ_W,ξ_{LM},ξ_{QLR} and their p-values . For all the three cases the asymptotic distribution is a χ_1^2 as there is only one restriction. By looking at the table two things emerge: first, that the tests are asymptotically equivalent as the t-stats and p-values are almost the same and second that they all don't reject the null hypothesis. Compared to part (a) p-values are lower, which means that the non rejection of the null is "stronger" than before.

	ξ	p-value
Wald	0.30953	0.57797
LM	0.29604	0.58638
QLR	0.29806	0.5851

Exercise 3

Part a

The following table sums up the results obtained in the previous problem set:

	$\hat{\delta}$	$\hat{\gamma}$	$\hat{CI_{0.05}}(\delta_0)$	$\hat{CI_{0.05}}(\gamma_0)$
GMM-two-step, 3 instruments	0.97853	3.386	$(0.93812, 1.0189)$	$(1.1577, 5.6142)$

Part b

To explain better the procedure I follow, I first report the entire code I use for this part.

```
% Part b: Block Bootstrap
% Routine elements
reps = 5000; %number of bootstrap repetitions
g_center=(sum(eul2')/T)';
reps_done=0;
tic
for k=[5 2 10];
for j=1:reps
data_b=block(data,k);
c_b = data_b(1,:);
d_b = data_b(2,:);
r_b = data_b(3,:);

% Balance the observations
c = c_b;
r = r_b;
d = d_b;
c_1=c(2:end);
```

```
r_1=r(2:end);
d_1=d(2:end);
r(250)=[];c(250)=[];d(250)=[];
T = length(c);
z=[ones(1,T); exp(r); (1+c)];

% Compute GMM - first step
W_1=eye(3);
fun1_b = @(x)objectivebs(x,g_center, W_1, z, T, c_1, r_1);
x0=[0.9 0.9];
options = optimset('MaxFunEvals',1000);
[theta1_gmm_b, Qval1_b] = fminsearch(fun1_b, x0,options);

% Compute GMM - second step
% Euler conditions at parameters estimated in step 1
eul_b = eebs(theta1_gmm_b,g_center, z, T, c_1, r_1);
% 2nd-step weighting matrix - the long-run variance, without lags (mds)
W_2_b=inv(longrunW(eul_b, T, 0));
fun2_b = @(x)objectivebs(x,g_center, W_2_b, z, T, c_1, r_1);
[theta2_gmm_b, Qval2_b]= fminsearch(fun2_b, theta1_gmm_b,options);

% Compute Asymptotic variance (G'S^(-1)G)^(-1) using consistent estimators
 G_b = [sum(exp(-theta2_gmm_b(2).*c_1+r_1))/T,
sum(-c_1.*theta2_gmm_b(1).*exp(-theta2_gmm_b(2).*c_1+r_1))/T;
 sum(exp(-theta2_gmm_b(2).*c_1+r_1+r))/T,
sum(-c_1.*theta2_gmm_b(1).*exp(-theta2_gmm_b(2).*c_1+r_1+r))/T;
 sum(exp(-theta2_gmm_b(2).*c_1+r_1).*(1+c))/T,
sum(-c_1.*theta2_gmm_b(1).*exp(-theta2_gmm_b(2).*c_1+r_1).*(1+c))/T];

eul2_b = eebs(theta2_gmm_b,g_center, z, T, c_1, r_1);
W_2_2 = inv(longrunW(eul2_b, T, 0));
AVar_b = inv(G_b'*W_2_2*G_b);

theta_bs(j,:)=theta2_gmm_b;
Tn_delta_bs(j,:) = (theta2_gmm_b(1)-theta2_gmm(1))/sqrt(AVar_b(1,1)/T);
Tn_gamma_bs(j,:) = (theta2_gmm_b(2)-theta2_gmm(2))/sqrt(AVar_b(2,2)/T);
if rem(j/200,1) == 0
j
end
end
eval(['theta_bs_',num2str(k),'=theta_bs;']);
eval(['Tn_delta_bs_',num2str(k),'=Tn_delta_bs;']);
eval(['Tn_gamma_bs_',num2str(k),'=Tn_gamma_bs;']);
end
toc
```

At the beginning of this part of the code I set up the routine elements: the number of bootstrap repetitions, that I choose equal to 5000 (i.e. $B = 5000$) in this case, g_{center} that is $\frac{1}{T}\sum_{t=1}^{n} g\left(w_t; \hat{\theta}_{GMM,2}\right)$ reported in part (a) as I have to maximize with respect to $g^*(w_t;\theta) = g(w_t;\theta) - g_{center}$ as the bootstrap targets the distribution under the two-step optimally weighted GMM reported at the previous point. I also set up a counter *reps_ done* to monitor the advancement of the procedure. Finally, I loop the whole procedure for all the block lenghts I have to use, i.e. 5, 2, 10 and for the number of bootstrap repetitions. Basically for each block lenght the program is going to compute 5000 estimates of $\hat{\theta}^*$ and of the t-stats for $\hat{\delta}^*$ and $\hat{\gamma}^*$. Each estimate is stored into *theta_ bs*, *Tn_ delta_ bs* and *Tn_ gamma_ bs* at the end of the bootstrap loop (these are therefore 5000x1 vectors). The following *if* statement takes track of the advancement of the procedure (it just reports the number of repetition if it is a multiple of 200). Finally, the last three commands *eval* rename the containers *theta_ bs*, *Tn_ delta_ bs* and *Tn_ gamma_ bs* according to the block length that is being looped before passing to the next value of the block that has to be analyzed. It takes 447 seconds for the code to be run for 5000 iterations.

(I)

To block bootstrap the original dataset I wrote the following function, named *block*:

```
%% BLOCK BOOTSTRAP %%
% This function performs block bootstrapping of data
% Reshuffling is done across columns

% INPUTS %
% x = data to be reshuffled
% blocks = block lenght

% OUTPUTS %
% x_b = reshuffled series

function x_b = block(x,blocks);
for i=1:blocks:(size(x,2)-blocks+1);
j=randi(size(x,2)-blocks+1);
x_b(:,i:i+blocks-1) = x(:,j:j+blocks-1);
end
```

Basically, this function takes as inputs a dataset structured in rows (e.g. in this case c, d, r series correspond to rows 1, 2, 3 respectively of the dataset) and the block number and returns a reshuffled dataset. The way the procedure works is best explained with an example. Suppose that the dataset has 249 columns and the block lenght is 5 (as in our baseline case). The function fixes the first column of the dataset (all rows are reshuffled), i.e. $i = 1$, then a random integer j between 1 and 249-5+1 (the last value should be consistent with the block lenght that has to be extracted) is extracted. Then a block 5 columns long starting from j is extracted and replaced in the bootstrapped series $x\_ b$ in the first 5 columns. This is repeated for all the successive i, distanced by the block length (in this case, the following i is 6) until the last column useful to insert a block (meaning the 249-5+1=245 column in this case).

In my code, this part correspond to the following:

```
data_b=block(data,k);
c_b = data_b(1,:);
d_b = data_b(2,:);
r_b = data_b(3,:);
```

I use the function *block* on the original dataset and then extract the reshuffled rows corresponding to the c, d, r series. Then I balance the dataset in the same way as the previous problem set. Note this is done for all the 5000 bootstrap iterations.

(II)

The 2-step efficient GMM estimator $\hat{\theta}^*$ in the 3-instruments case is computed in this part:

```
z=[ones(1,T); exp(r); (1+c)];

% Compute GMM - first step
W_1=eye(3);
fun1_b = @(x)objectivebs(x,g_center, W_1, z, T, c_1, r_1);
x0=[0.9 0.9];
options = optimset('MaxFunEvals',1000);
[theta1_gmm_b, Qval1_b] = fminsearch(fun1_b, x0,options);

% Compute GMM - second step
% Euler conditions at parameters estimated in step 1
eul_b = eebs(theta1_gmm_b,g_center, z, T, c_1, r_1);
% 2nd-step weighting matrix - the long-run variance, without lags (mds)
W_2_b=inv(longrunW(eul_b, T, 0));
fun2_b = @(x)objectivebs(x,g_center, W_2_b, z, T, c_1, r_1);
[theta2_gmm_b, Qval2_b]= fminsearch(fun2_b, theta1_gmm_b,options);

% Compute Asymptotic variance (G'S^(-1)G)^(-1) using consistent estimators
 G_b = [sum(exp(-theta2_gmm_b(2).*c_1+r_1))/T,
sum(-c_1.*theta2_gmm_b(1).*exp(-theta2_gmm_b(2).*c_1+r_1))/T;
 sum(exp(-theta2_gmm_b(2).*c_1+r_1+r))/T,
sum(-c_1.*theta2_gmm_b(1).*exp(-theta2_gmm_b(2).*c_1+r_1+r))/T;
 sum(exp(-theta2_gmm_b(2).*c_1+r_1).*(1+c))/T,
sum(-c_1.*theta2_gmm_b(1).*exp(-theta2_gmm_b(2).*c_1+r_1).*(1+c))/T];

eul2_b = eebs(theta2_gmm_b,g_center, z, T, c_1, r_1);
W_2_2 = inv(longrunW(eul2_b, T, 0));
AVar_b = inv(G_b'*W_2_2*G_b);
```

The procedure is basically the usual one, except that the objective function and the variance should be computed with respect to the recentered moment function $g^*(w_t;\theta) = g(w_t;\theta) - g_{center}$. Therefore I have adapted the functions *objectivebs* and *eebs* (respectively, the function that computes the objective function and the one that computes the Euler conditions under a specific value of θ in order to put this series into the function *longrunW* that computes the variance - there are zero lags as the process is a martingale difference sequence). Then the asymptotic variance *AVar_ b* in the last line of the part of code reported above is computed by calculating the matrix of derivatives G and the two-step optimal variance using $\hat{\theta}^*$ computed in the second step. I report *objectivebs* and *eebs* below:

```
%% Set the objective function for bootstrap (re-centered)
% This program computes the objective function for Euler Equation conditions
%%INPUTS%%
% x = (delta,gamma) parameters to be estimated
% W = weighting matrix (KxK)
% z = observables at t
% T=obs number
% c_1= log(C_t+1/C_t)
% c= log(C_t/C_t-1), r_1= log(R_t+1)
% g_center=it is the average of g computed at two-step GMM
%%OUTPUTS%%
% Sdf = stochastic discount factor (1xT)
% Euler = set of euler conditions equal to zero (K*T)
% g = sample average of each Euler (Kx1)
% Qn = objective function to minimize (1x1)
% Euler_bs = Euler recentered
% default output is Qn only
function Qn = objectivebs(x,g_center, W, z, T, c_1, r_1);
euler=(x(1)*exp(-x(2)*c_1+r_1)-1).*z;
euler_bs=euler-g_center;
g=[sum(euler_bs')/T]';
Qn = 0.5*(g'*W*g);
end
```

```
%% Euler conditions Bootstrap
% This program computes Euler conditions given parameters
%%INPUTS%%
% x = (delta,gamma) estimated parameters
% z = observables at t
% T=obs number
% c_1= log(C_t+1/C_t)
% c= log(C_t/C_t-1), r_1= log(R_t+1)
% g_center=it is the average of g computed at two-step GMM
```

```
%%OUTPUTS%%
% sdf = stochastic discount factor (1xT)
% euler = set of euler conditions equal to zero (K*T) - default only ee
% Euler_bs = Euler recentered
function euler_bs = eebs(x,g_center, z, T, c_1, r_1);
euler=(x(1)*exp(-x(2)*c_1+r_1)-1).*z;
euler_bs=euler-g_center;
end
```

(III)

The following part of the code stores the estimated obtained in each bootstrap repetition. In particular, for each iteration, the first line stores $\hat{\theta}^*$ and the second the t-statistics:

$$T^*_{n,\delta} = \frac{\hat{\delta}^* - \hat{\delta}_{GMM,2}}{\sqrt{\frac{\hat{\Omega}^*_{1,1}}{T}}} \qquad T^*_{n,\gamma} = \frac{\hat{\gamma}^* - \hat{\gamma}_{GMM,2}}{\sqrt{\frac{\hat{\Omega}^*_{2,2}}{T}}}$$

Where $\hat{\Omega}^*_{1,1}$ and $\hat{\Omega}^*_{2,2}$ are the diagonal elements of the bootstrap asymptotic variance. I don't take here the absolute value in order to check the the asymptotic distribution of the two t-stats. I take the absolute value afterwards to compute the quantiles and the CI.

```
theta_bs(j,:)=theta2_gmm_b;
Tn_delta_bs(j,:) = (theta2_gmm_b(1)-theta2_gmm(1))/sqrt(AVar_b(1,1)/T);
Tn_gamma_bs(j,:) = (theta2_gmm_b(2)-theta2_gmm(2))/sqrt(AVar_b(2,2)/T);
```

Figure 1 plots the distributions of the two t-stats for all the block lengths. In all the cases they look standard normals centered at zero and they are almost symmetric except there are some asymmetries in the left tails.

(IV)

As $H_n\left(z, \hat{P}\right)$ is consistent for $H_n\left(z, P_0\right)$, we can construct CI for the two parameters as follows:

$$\hat{\delta}_{GMM,2} \pm z^*_{\frac{\alpha}{2},\delta}\sqrt{\frac{\hat{\Omega}_{1,1}}{T}} \qquad \hat{\gamma}_{GMM,2} \pm z^*_{\frac{\alpha}{2},\gamma}\sqrt{\frac{\hat{\Omega}_{2,2}}{T}}$$

Where $\hat{\Omega}_{1,1}$ and $\hat{\Omega}_{2,2}$ are the diagonal elements of the two-step efficient GMM asymptotic variance and $z^*_{\frac{\alpha}{2},\delta}$ and $z^*_{\frac{\alpha}{2},\gamma}$ are respectively the $1-\alpha$ quantiles of the distribution of $\left|T^*_{n,\delta}\right|$ and $\left|T^*_{n,\gamma}\right|$.

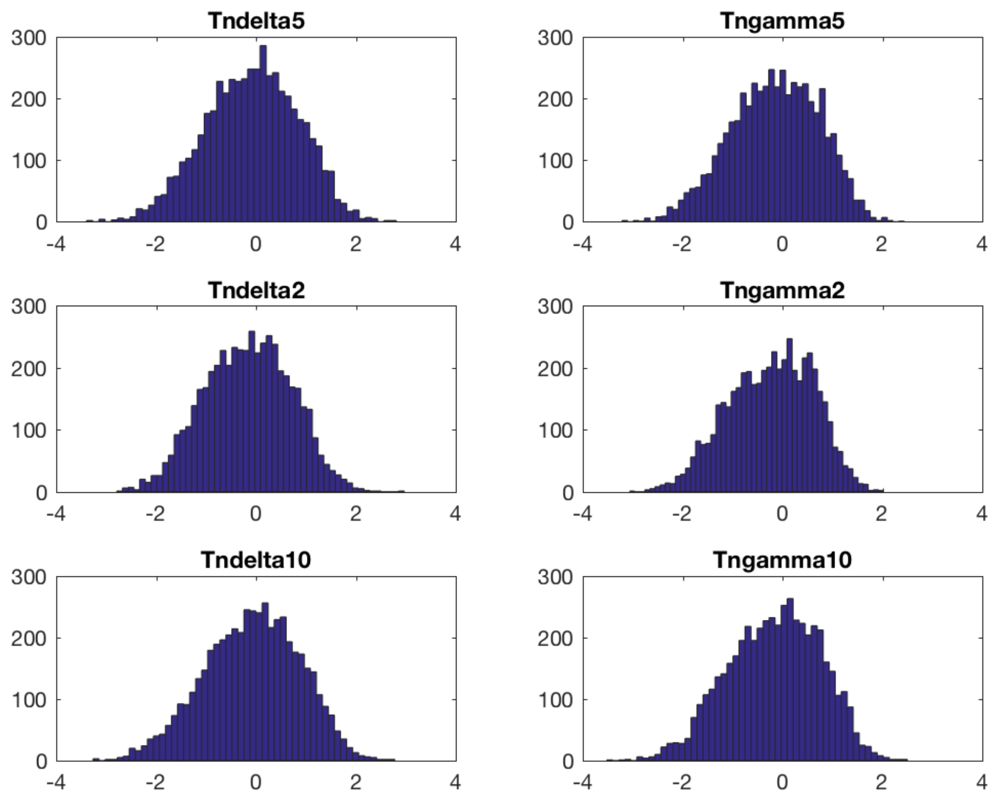
The table below reports the confidence intervals for the parameters δ_0 and γ_0 (the code is in part (c) and (d) as I did it for all block lengths at the same time).

	$100\,(1-0.05)\,\%\ \hat{CI}(\delta_0)$	$100\,(1-0.05)\,\%\hat{CI}(\gamma_0)$
block length $= 5$	$(0.94206, 1.015)$	$(1.4287, 5.3433)$

Part c

Using the steps of part (b) for all the block lenghts I compute the 95% CI (note that in the code I use $1-\alpha/2$ as I choose $\alpha = 0.1$) with the following code according to the formula pointed out at point

Figure 1: Distributions of parameters' T-stats for different block lenghts, 5000 bootstrap repetitions (not in absolute value)



(b-IV):

```
%Bootstrap Confidence intervals
alpha=0.10;
delta_CI_bs_5=[theta2_gmm(1)-quantile(abs(Tn_delta_bs_5),1-alpha/2)*
sqrt(AVar(1,1)/T)...
theta2_gmm(1)+quantile(abs(Tn_delta_bs_5),1-alpha/2)*sqrt(AVar(1,1)/T)];
delta_CI_bs_2=[theta2_gmm(1)-quantile(abs(Tn_delta_bs_2),1-alpha/2)*
sqrt(AVar(1,1)/T)...
theta2_gmm(1)+quantile(abs(Tn_delta_bs_2),1-alpha/2)*sqrt(AVar(1,1)/T)];
delta_CI_bs_10=[theta2_gmm(1)-quantile(abs(Tn_delta_bs_10),1-alpha/2)*
sqrt(AVar(1,1)/T)...
theta2_gmm(1)+quantile(abs(Tn_delta_bs_10),1-alpha/2)*sqrt(AVar(1,1)/T)];
gamma_CI_bs_5=[theta2_gmm(2)-quantile(abs(Tn_gamma_bs_5),1-alpha/2)*
sqrt(AVar(2,2)/T)...
theta2_gmm(2)+quantile(abs(Tn_gamma_bs_5),1-alpha/2)*sqrt(AVar(2,2)/T)];
gamma_CI_bs_2=[theta2_gmm(2)-quantile(abs(Tn_gamma_bs_2),1-alpha/2)*
sqrt(AVar(2,2)/T)...
theta2_gmm(2)+quantile(abs(Tn_gamma_bs_2),1-alpha/2)*sqrt(AVar(2,2)/T)];
gamma_CI_bs_10=[theta2_gmm(2)-quantile(abs(Tn_gamma_bs_10),1-alpha/2)*
sqrt(AVar(2,2)/T)...
theta2_gmm(2)+quantile(abs(Tn_gamma_bs_10),1-alpha/2)*sqrt(AVar(2,2)/T)];

delta_CI_bs=table(delta_CI_bs_5,delta_CI_bs_2,delta_CI_bs_10)
gamma_CI_bs=table(gamma_CI_bs_5,gamma_CI_bs_2,gamma_CI_bs_10)
```

The table below reports the 95% CI for the parameters for all the block length specifications and for the two-step GMM:

	$100\,(1-0.05)\,\%\ \hat{CI}(\delta_0)$	$100\,(1-0.05)\,\%\hat{CI}(\gamma_0)$
block length = 2	$(0.9441, 1.013)$	$(1.4634, 5.3086)$
block length = 5	$(0.94206, 1.015)$	$(1.4287, 5.3433)$
block length = 10	$(0.94092, 1.0161)$	$(1.3883, 5.3836)$
GMM	$(0.93812, 1.0189)$	$(1.1577, 5.6142)$

From the table it clearly emerges that the CI intervals for δ are similar among all the cases while the results are more sensitive for γ (as the latter estimate suffers from the higher variance of γ). Therefore, results are in general sensitive to the choice of block length and the problem enlarges when the parameter is less accurately estimated. Moreover, it is worth to notice that the left bound of both CI decreases with the number of block length while the right bound increases with the number of block length. In both cases the case with block length equal to 10 leads to CI that are the nearest to the GMM case (we cannot conclude anything from this because maybe the asymptotic distribution of the GMM was not accurate for our sample size). We know that for the block bootstrap to be valid we should have that the block lenght L converges to infinity as T goes to infinity faster than T. This implies that $L = O\left(T^{\alpha}\right)$ and from the notes we know that typically $\alpha \in [0, \frac{1}{3}]$ and that some results on the optimal choice of L suggest $L = O\left(T^{1/4}\right)$ or $L = O\left(T^{1/5}\right)$. In our context $T = 249$ which means that:

$$L = \begin{cases} [O(1), O(6.29)] & if \quad \alpha \in [0, \frac{1}{3}] \\ O(3.97) & if \quad \alpha = 1/4 \\ O(3.01) & if \quad \alpha = 1/5 \end{cases}$$

Therefore, according to the above discussion, the procedure should be more accurate when $L = \{2, 5\}$. In fact, confidence intervals in these two cases are more precise being more narrow than when $L = 10$. Intuitively this is what we should expect as with lower lenght there should be "more" reshuffling. However, this may also be due to the fact that the distribution we get have a slightly bigger left tail, which means they are not completely symmetric, which in turn leads (as described in part (d)) to critical values that are wrong as the quantiles are not symmetric.

Part d

I use the following code to obtain the critical values $z^*_{\frac{\alpha}{2},\delta}$ and $z^*_{\frac{\alpha}{2},\gamma}$ (i.e. the $1-\alpha$ quantiles of the distribution of the absolute value of the t-stats):

```
quantiles_delta=table(quantile(abs(Tn_delta_bs_5),1-alpha/2),
quantile(abs(Tn_delta_bs_2),1-alpha/2),quantile(abs(Tn_delta_bs_10),1-alpha/2))

quantiles_gamma=table(quantile(abs(Tn_gamma_bs_5),1-alpha/2),
quantile(abs(Tn_gamma_bs_2),1-alpha/2),quantile(abs(Tn_gamma_bs_10),1-alpha/2))
```

The table below reports the critical values obtained:

	$z^*_{\frac{\alpha}{2},\delta}$	$z^*_{\frac{\alpha}{2},\gamma}$
block length = 2	1.6697	1.6911
block length = 5	1.7686	1.7216
block length = 10	1.8241	1.7572

It is clear that these values are different from 1.96. Probably there is some asymmetry in the distribution of the bootstrap (indeed Figure 1 reveals that the distributions are a little bit more balanced on the left tail), and this issue seems more relevant as the block length decreases. Probably this is because the sampling distribution is actually not exactly a standard normal.