

Quantitative Macroeconomics

Problem Set 6
November 15, 2018

Mridula Duggal

A Initial values and exogenous process.

A.1 Initial guess.

We have to come up with initial guess for different value functions of the type $v(k, \epsilon; \bar{k}, z_s)$ for $\epsilon = \{0, 1\}$ and $z = \{good, bad\}$ assuming that:

- Agents expect that the individual and aggregate states (\bar{k}, ϵ, z) will not change in the future.
- Agents' policy is such that $g(k, \epsilon; \bar{k}, z) = k$

The value function boils down to the following expression:

$$v(k, \epsilon; \bar{k}, z_s) = \max_{\{c, k'\}} \frac{c^{1-\gamma} - 1}{1 - \gamma} + \beta \mathbb{E}[v(k', \epsilon'; \bar{k}', z'_s)] \quad (1)$$

where

$$c = w(\bar{k}, z_s)\epsilon + (1 + r(\bar{k}, z_s) - \delta)k - g(\bar{k}, z_s) \quad (2)$$

The technology is given by a Cobb-Douglas aggregate production function:

$$y = z\bar{k}^\alpha l^{(1-\alpha)} \quad (3)$$

Then, from (3) we can recover factor prices:

$$w(\bar{k}, z_s) = (1 - \alpha)z_s \left(\frac{\bar{k}}{l}\right)^\alpha \quad (4)$$

$$r(\bar{k}, z_s) = \alpha z_s \left(\frac{l}{\bar{k}}\right)^{1-\alpha} \quad (5)$$

Now, plug (4) and (5) in (2) and (2) in (1):

$$v(k, \epsilon; \bar{k}, z_s) = \max_{\{k'\}} \frac{\left[\left((1 - \alpha)z_s \left(\frac{\bar{k}}{l}\right)^\alpha \right) \epsilon + \left(1 + \alpha z_s \left(\frac{l}{\bar{k}}\right)^{1-\alpha} - \delta \right) k - g(\bar{k}, z_s) \right]^{1-\gamma} - 1}{1 - \gamma} + \beta \mathbb{E}[v(k', \epsilon'; \bar{k}', z'_s)] \quad (6)$$

Given our assumptions:

- $g(\cdot) = k$.
- $\mathbb{E}[v(k', \epsilon'; \bar{k}', z'_s)] = v(k, \epsilon; \bar{k}, z_s)$

(6) reduces to:

$$v(k, \epsilon; \bar{k}, z_s) = \frac{\left[\left((1 - \alpha)z_s \left(\frac{\bar{k}}{l}\right)^\alpha \right) \epsilon + \left(\alpha z_s \left(\frac{l}{\bar{k}}\right)^{1-\alpha} - \delta \right) k \right]^{1-\gamma} - 1}{(1 - \gamma)(1 - \beta)} \quad (7)$$

From (7) we can derive expressions for the value functions depending on both the idiosyncratic and aggregate shock:

$$v(k, 1; \bar{k}, z_g) = \frac{[(1 - \alpha)z_g(\frac{\bar{k}}{1 - u_g})^\alpha + (\alpha z_g(\frac{1 - u_g}{\bar{k}})^{1 - \alpha} - \delta)k]^{1 - \gamma} - 1}{(1 - \gamma)(1 - \beta)} \quad (8)$$

$$v(k, 0; \bar{k}, z_g) = \frac{[(\alpha z_g(\frac{1 - u_b}{\bar{k}})^{1 - \alpha} - \delta)k]^{1 - \gamma} - 1}{(1 - \gamma)(1 - \beta)} \quad (9)$$

$$v(k, 1; \bar{k}, z_b) = \frac{[(1 - \alpha)z_b(\frac{\bar{k}}{1 - u_g})^\alpha + (\alpha z_b(\frac{1 - u_g}{\bar{k}})^{1 - \alpha} - \delta)k]^{1 - \gamma} - 1}{(1 - \gamma)(1 - \beta)} \quad (10)$$

$$v(k, 0; \bar{k}, z_b) = \frac{[(\alpha z_b(\frac{1 - u_b}{\bar{k}})^{1 - \alpha} - \delta)k]^{1 - \gamma} - 1}{(1 - \gamma)(1 - \beta)} \quad (11)$$

A.2 Transition matrix

We have two idiosyncratic shocks $\epsilon = \{0, 1\}$ and two aggregate shocks $z = \{bad, good\}$. Thus, the economy can be shocked by 4 different types of shocks $\{0b, 1b, 0g, 1g\}$. As a result, we need to build a 4x4 transition matrix. For doing so, we have used a system of 16 equations, coming from rows properties (equal to 1), the information about the duration of good and bad times, the duration of unemployment and so on. See the Python code for the details. In the end, we got the following transition matrix. Note that first row/column is for 0b, second for 1b, third 0g and fourth 1g. Lastly, notice that the higher probabilities are in the diagonal, meaning that there is a high degree of persistence:

$$\Pi = \begin{bmatrix} 0.6950 & 0.1800 & 0.0809 & 0.0441 \\ 0.0200 & 0.8550 & 0.0049 & 0.1200 \\ 0.1241 & 0.0009 & 0.7550 & 0.1200 \\ 0.0078 & 0.1171 & 0.0050 & 0.8700 \end{bmatrix}$$

B Workers problem and simulation

B.1 Solution of the model by VFI.

We have tried to solve it with Python but we got flat policy functions. It seems that the budget constraint is not working properly, but we have not figured it out why yet. As an alternative, we have built a Matlab code (based on the sample code provided by the professor). By doing so, we have got the following policy functions.

Figure 1: Assets policy function for a good productivity shock.

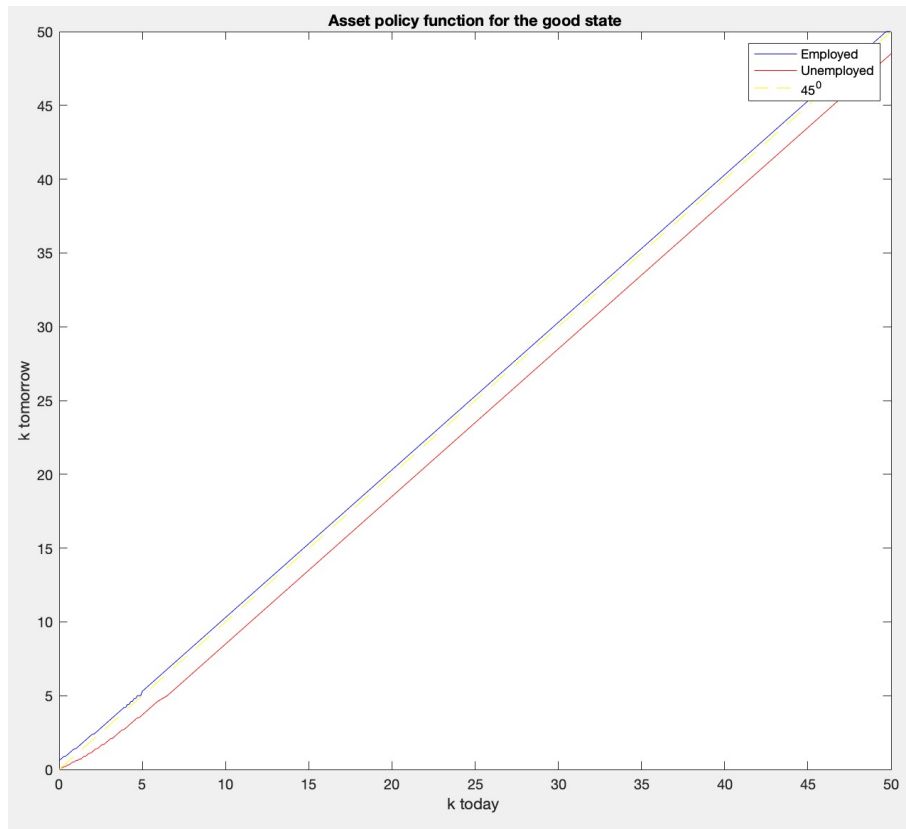
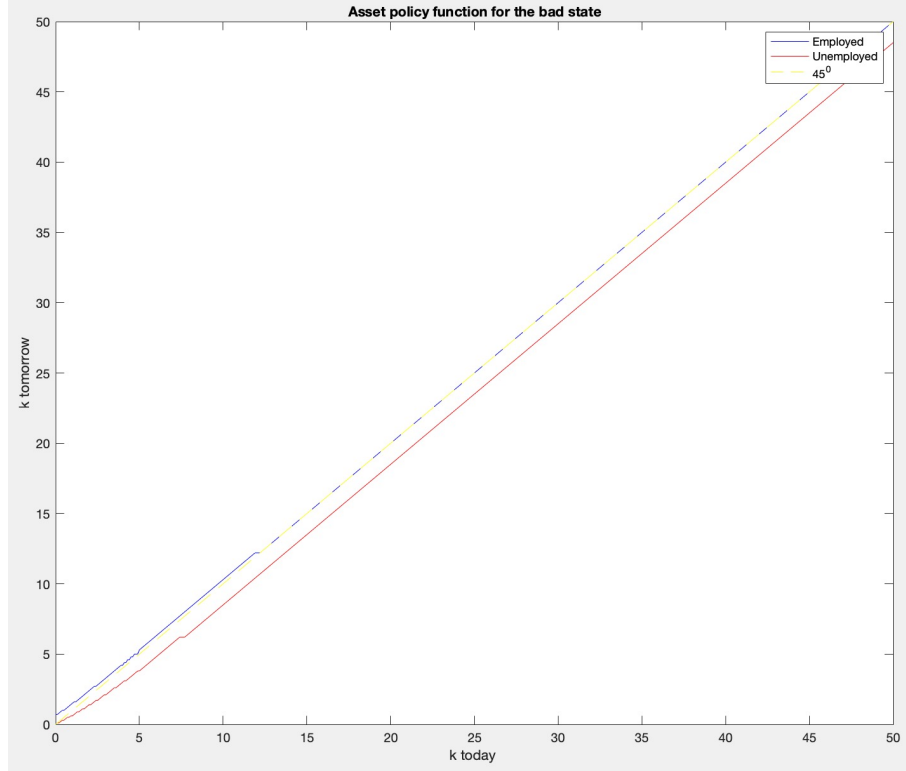


Figure 2: Assets policy function for a bad productivity shock.



When the economy is in good times, the employed agent is always accumulating assets (saving) and the unemployed guy is depleting assets. This agrees with the common sense: you save when employed and dis-save (or borrow) when unemployed. When bad times come, at some point the employed guy stops saving and just keep balance her budget. On the contrary, if a guy is shocked enough times with the unemployment, she will ends up having no assets (poverty trap), and will be borrowed constraint.

B.2 Simulation for 1000 agents and 2000 periods.

See the Matlab code.

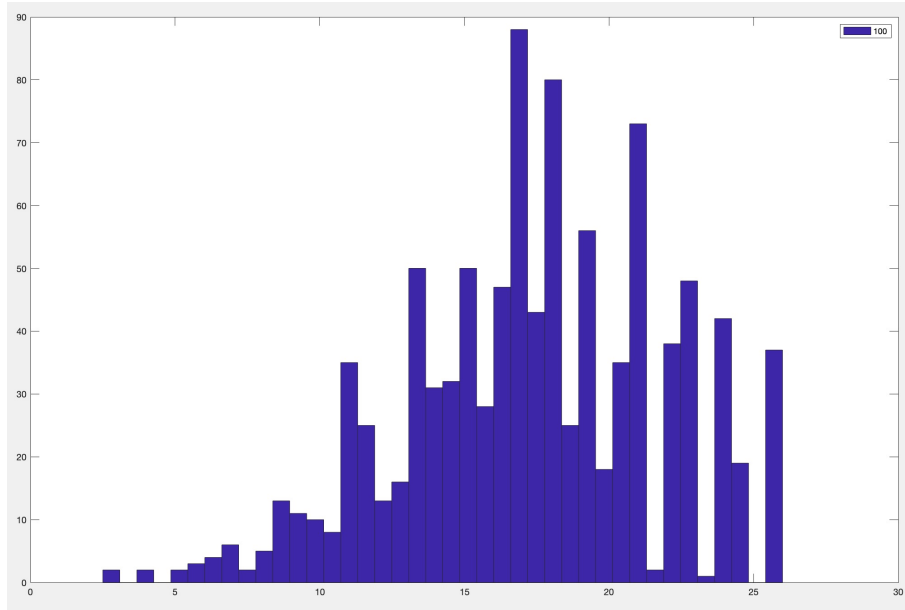
C Solution of the model

C.1 Parameter values and distribution.

For the procedure to get convergence in β 's see the Matlab code. Basically, we estimate the parametrized expectations parameters by using the time series for K and z got in the previous exercise; then we update the β in the function H; simulate the model again; get time series for K and z ; estimate β ; keep doing that until the new β were almost equal to the previous one (convergence). The goodness of the estimation is high (R^2 higher than 0.9 for both states).

Now, let's take a look at the equilibrium asset distribution. Figure 3 plot it. The average capital, which in this setup is equal to the aggregate capital is 18.28. The standard deviation is 4.75, and the skewness is 0.01. It means that the inequality is quite high and that there are more people owning a high level of assets than poor.

Figure 3: Equilibrium asset distribution.



Now, we have run the experiment of comparing the evolution of the stationary distribution after 7 consecutive periods of being in a bad state (graph 4) and 7 consecutive periods of being in a good one (graph 5). For bad periods, the sd is 5.93 and the skewness is 1.45. For good periods, the sd is 5.09 and the skewness is 0.09. Then, for bad times the inequality is higher than for good, and also there are more people having more assets.

Figure 4: Equilibrium asset distribution after 7 bad periods.

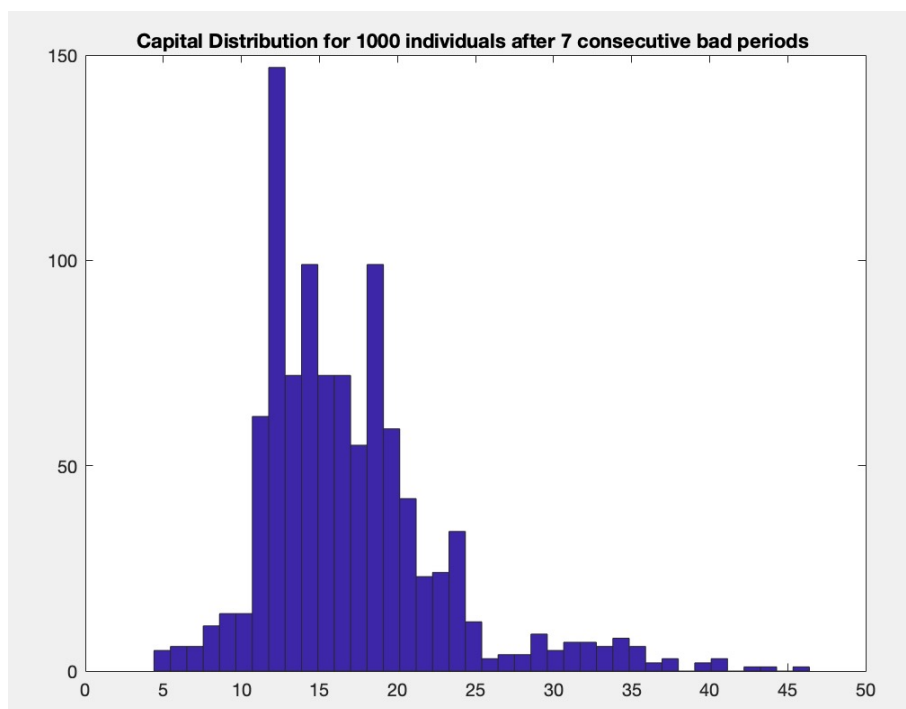
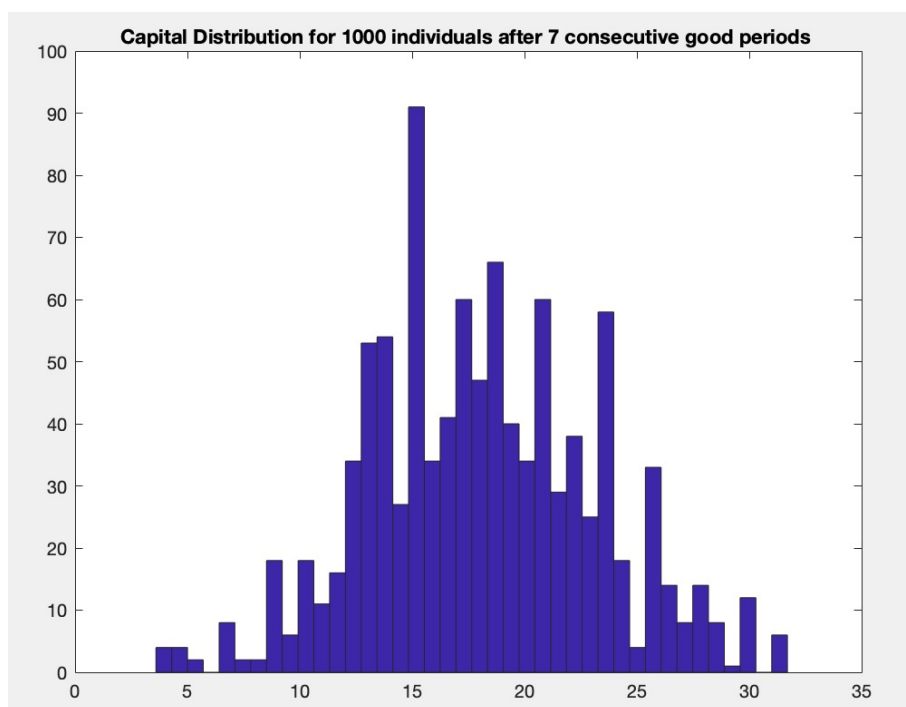


Figure 5: Equilibrium asset distribution after 7 good periods.



C.2 MATLAB Code - Baseline Kruseel-Smith

```

1 %% Code for solving a simplified version of K&S (1998)
2 % Quantitative Macroeconomics – IDEA programme
3
4 %% Initial values and parameters
5
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Finding the transition matrix for the state
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8 % The system of equations
9 A= [ 1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 ; ...
10      0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0 ; ...
11      0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0 ; ...
12      0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1 ; ...
13      0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0 ; ...
14      0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0 ; ...
15      0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0 ; ...
16      0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0 ; ...
17      1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ; ...
18      0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 ; ...
19      0  0  0  0  0  0  0  0  0  5.6  0  -1  0  0  0  0 ; ...
20     -1  0  28/3  0  0  0  0  0  0  0  0  0  0  0  0  0 ; ...
21     .02 .48 .05 .45 0  0  0  0  0  0  0  0  0  0  0  0 ; ...
22      0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ; ...
23      0  0  0  0  0  0  0  0 .02 .48 .05 .45 0  0  0  0 ; ...
24      0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 ];
25
26
27 b= [7/8; 7/8; 7/8; 7/8; 1/8; 1/8; 1/8; 1/8; 7/24; 21/40; 0; 0; 0.02;
     0.005; 0.05; 0.02];
28
29
30 pize = reshape(A^-1*b,4,4);
31
32
33
34 % transtion matrix aggregate state
35
36 piZ = [ 7/8  1/8;...
37         1/8  7/8];
38
39
40 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41

```



```

42  betta=0.95;
43  delta=0.0025;
44  z=[1.01  0.99];
45  alfa=0.36;
46  L=[0.96,  0.9];
47
48  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Starting values for V %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49
50  v1g = @(k,K) log( alfa*z(1)*(K/L(1))^(alfa-1)*k+ (1-alfa)*z(1)*(K/L
    (1))^(alfa) -delta*k )/(1-betta);
51  v1b = @(k,K) log( alfa*z(2)*(K/L(2))^(alfa-1)*k+ (1-alfa)*z(2)*(K/L
    (2))^(alfa) -delta*k )/(1-betta);
52  v0g = @(k,K) log( alfa*z(1)*(K/L(1))^(alfa-1)*k -delta*k )/(1-betta);
53  v0b = @(k,K) log( alfa*z(2)*(K/L(2))^(alfa-1)*k -delta*k )/(1-betta);
54
55
56  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Grid for k and K %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57
58  k_grid=[0:0.1:5 ,5.3:0.3:50];
59  K_grid=[16:0.04:18.5];
60
61  % Evaluation of the VF
62  for j=1:size(K_grid,2)
63  V1g(:,j)= v1g(k_grid ,K_grid(j)) ';
64  V1b(:,j)= v1b(k_grid ,K_grid(j)) ';
65  V0g(:,j)= v0g(k_grid ,K_grid(j)) ';
66  V0b(:,j)= v0b(k_grid ,K_grid(j)) ';
67  end
68
69
70
71  %%
72  %%%%%%%%% Perceived law of motion %%%%%%%%%
73  % initial values
74
75  b0g=0;
76  b1g=1;
77  b0b=0;
78  b1b=1;
79
80  %%
81  for iter_b=1:1000
82  iter_b
83  % zi is the index for good shock

```

```

84 H=@(K, zi) exp( (b0g+b1g*log(K))*zi+ (b0b+b1b*log(K))*(1-zi) );
85
86 % approximation
87
88 Ha= @(K, zi) min(abs(K_grid-H(K, zi)));
89
90
91
92
93 %% Solution of the consumer problem
94
95
96 % Consumption for each possible decision
97
98 % e=1 employed
99 % g=1 good times =2 bad times
100 c= @(i, I, e, g) max( alfa*z(g)*(K_grid(I)/L(g))^( alfa-1).*k_grid(i)+ ...
101 (1- alfa)*z(g)*(K_grid(I)/L(g))^( alfa)*e +(1-delta)*
102 k_grid(i) ...
103 - k_grid,0) ;
104
105 for iter=1:1000
106 for i=1:size(k_grid,2)
107 for I=1:size(K_grid,2)
108
109 % approximation next period capital
110
111 [ dif, Ip]=min(abs(K_grid-H(K_grid(I),1)));
112 V0gt(i, I)= max(log(c(i, I, 0, 1))' + betta * ([ pize(1,:) ]*( [V0g
113 (:, Ip), V1g(:, Ip), V0b(:, Ip), V1b(:, Ip)] ')) '));
114 V1gt(i, I)= max(log(c(i, I, 1, 1))' + betta * ([ pize(2,:) ]*( [V0g
115 (:, Ip), V1g(:, Ip), V0b(:, Ip), V1b(:, Ip)] ')) '));
116
117 [ dif, Ip]=min(abs(K_grid-H(K_grid(I),0)));
118 V0bt(i, I)= max(log(c(i, I, 0, 2))' + betta * ([ pize(3,:) ]*( [V0g
119 (:, Ip), V1g(:, Ip), V0b(:, Ip), V1b(:, Ip)] ')) '));
120 V1bt(i, I)= max(log(c(i, I, 1, 2))' + betta * ([ pize(4,:) ]*( [V0g
121 (:, Ip), V1g(:, Ip), V0b(:, Ip), V1b(:, Ip)] ')) '));
122
123 end
124 end

```

```

123 dev= max(max(abs( [V0gt-V0g,V1gt-V1g,V0bt-V0b,V1bt-V1b] ) ) );
124
125 if dev<0.00001
126     break
127 else
128     V0g=V0gt;
129     V1g=V1gt;
130     V0b=V0bt;
131     V1b=V1bt;
132 end
133
134 end
135
136
137 % Recover the policy function
138
139
140 for i=1:size(k_grid,2)
141     for I=1:size(K_grid,2)
142
143         % approximation next period capital
144
145         [dif,Ip]=min(abs(K_grid-H(K_grid(I),1)));
146         [V0gt(i,I),a(i,I,2,1)]= max(log(c(i,I,0,1))' + betta * ([pize
147             (1,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))');
148         [V1gt(i,I),a(i,I,1,1)]= max(log(c(i,I,1,1))' + betta * ([pize
149             (2,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))');
150
151         [dif,Ip]=min(abs(K_grid-H(K_grid(I),0)));
152         [V0bt(i,I),a(i,I,2,2)]= max(log(c(i,I,0,2))' + betta * ([pize
153             (3,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))');
154         [V1bt(i,I),a(i,I,1,2)]= max(log(c(i,I,1,2))' + betta * ([pize
155             (4,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))');
156
157     end
158 end
159
160 %Store the policy functions.
161 assets(:,:,2,1) = a(i,I,2,1); %employed, bad
162 assets(:,:,1,1)= a(i,I,1,1); % unemployed, bad
163 assets(:,:,2,2) = a(i,I,2,2); % employed, good
164 assets(:,:,1,2)= a(i,I,1,2); %unemployed, good

```

```

163 %% Simulation
164
165
166 % A sequence of TFP
167 % using the index =1 good ,   =2 bad
168
169 if iter_b==1
170
171   zt(1)=1;
172
173   for t=2:2000
174       draw=rand;
175       zt(t)= 1+(rand>=piZ(zt(t-1),1));
176   end
177 % Splitting the sample for good and bad times
178
179 % "burning" the first 200 periods
180 ztb=zt;
181 ztb(1:200)=0;
182 % Construct an index for the good times
183 i_zg=find(zt==1);
184
185 % Construct an index for the bad times
186 i_zb=find(zt==2);
187
188 % initial distribution of assets and employment
189 % =1 employed
190 N_state(1:960,:,1)=ones(960,1)*[26,1];
191 % =2 unemployed
192 N_state(961:1000,:,1)=ones(40,1)*[26,2];
193
194 K_sim(1)=find(K_grid == 17); %The index is for the generated capital
    from the simulated series.
195
196 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Simulating the Distributions
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
197
198 for t=2:2000
199     for n=1:1000
200
201     % Evolution of assets
202         N_state(n,1,t)=a(N_state(n,1,t-1),K_sim(t-1),N_state(n,2,t-1),zt(
            t-1));
203     % Evolution of the employment status

```

```

204     N_state(n,2,t)= 2-(rand>=pize(1 + zt(t-1)*2 - N_state(n,2,t-1),zt
      (t)*2-1)/piZ(zt(t-1),zt(t)));
205
206
207 end
208
209 % Storage of the sequence of aggregate capital
210 [dev2, K_sim(t)]=min(abs(k_grid(round(mean(N_state(:,1,t)))-K_grid))
      ;
211
212
213 end
214
215 else
216
217
218 for t=2:2000
219 for n=1:1000
220
221 % Evolution of assets
222     N_state(n,1,t)=a(N_state(n,1,t-1),K_sim(t-1),N_state(n,2,t-1),zt(
      t-1));
223
224 end
225
226 % Storage of the sequence of aggregate capital
227 [dev2, K_sim(t)]=min(abs(k_grid(round(mean(N_state(:,1,t)))-K_grid))
      ;
228
229
230 end
231
232 end
233
234 % Regression model for the evolution of aggregate capital
235
236 % regression for good times (burning the first 20 periods of g times)
237
238 Yg=log(K_grid(K_sim(i_zg(20:end)))');
239 Xg=[ones(size(i_zg(20:end),2),1),log(K_grid(K_sim(i_zg(20:end))-1))')]
      ;
240 Bg=Xg\Yg
241 b0gp=Bg(1);
242 b1gp=Bg(2);

```

```

243 % regression for bad times (burning the first 20 periods of bad times
    .
244
245 Yb=log ( K_grid ( K_sim ( i_zb ( 20:end ) ) ) ' ) ;
246 Xb=[ones ( size ( i_zb ( 20:end ) , 2 ) , 1 ) , log ( K_grid ( K_sim ( i_zb ( 20:end ) - 1 ) ) ' ) ]
    ;
247 Bb=Xb\Yb
248 b0bp=Bb(1) ;
249 b1bp=Bb(2) ;
250
251
252 dev_b=max( abs ( [ b0g-b0gp b1g-b1gp b0b-b0bp b1b-b1bp ] ) )
253
254 pause ( 1 )
255 if dev_b <= 0.01
256     break
257 end
258
259 b0g=0.1*b0gp+0.9*b0g ;
260 b1g=0.1*b1gp+0.9*b1g ;
261 b0b=0.1*b0bp+0.9*b0b ;
262 b1b=0.1*b1bp+0.9*b1b ;
263
264
265 end
266
267 %% Number of Iterations
268 disp ( 'Number of Iterations ' )
269 disp ( iter_b )
270
271 %% Mean of the Simulated series
272 mean_s = mean ( K_grid ( K_sim ) ) ;
273 disp ( 'Simulated series Aggregate/Average Capital is ' )
274 disp ( mean_s )
275
276 %% R2 for the regression
277
278 mdlg = fitlm ( Xg ( : , 2 ) , Yg ) ;
279 R2g = mdlg . Rsquared . Ordinary ;
280 stdg = mdlg . RMSE ;
281 disp ( 'R2 for the good shock ' )
282 disp ( R2g )
283
284 mdlb = fitlm ( Xb ( : , 2 ) , Yb ) ;

```

```

285 R2b = mdlb.Rsquared.Ordinary;
286 stdb= mdlb.RMSE;
287 disp('R2 for the bad shock')
288 disp(R2b)
289 %% Graphs for the Policy Functions and Asset Distribution
290
291 % Evolution of the Assets distribution
292 %Figure
293 for t_ind=1:100
294
295     hist(k_grid(reshape(N_state(:,1,t_ind),1,1000)),40)
296     legend(num2str(t_ind))
297     pause(1)
298
299 end
300
301 %% Statistics for the distriburion
302 W = k_grid(reshape(N_state(:,1,t_ind),1,1000));
303
304 mean_w = mean(W);
305 disp('Mean of Asset Distribution')
306 disp(mean_w)
307
308 std_w = std(W);
309 disp('Standard Deviation of Asset Distribution')
310 disp(std_w)
311
312 skew_w=skewness(W);
313 disp('Skewness of Asset Distribution');
314 disp(skew_w)
315
316 %% Retrieve the Policy function from the indices.
317
318 G0g = k_grid(a(:,26,2,1)); % unemployed, good
319 G1g = k_grid(a(:,26,1,1)); % employed, good
320 G0b = k_grid(a(:,26,2,2)); % unemployed, bad
321 G1b = k_grid(a(:,26,1,2)); % employed, bad
322
323 figure(1)
324 plot(k_grid, G1b, 'b' )
325 hold on
326 plot(k_grid, G0b, 'r' )
327 hold on
328 plot(k_grid, k_grid, 'y—')

```

```

329 title ( 'Asset policy function for the bad state' )
330 legend ( 'Employed' , 'Unemployed' , '45^0' )
331 xlabel ( 'k today ' )
332 ylabel ( 'k tomorrow' )
333 hold off
334
335 figure (2)
336 plot (k_grid , G1g , 'b' )
337 hold on
338 plot (k_grid , G0g , 'r' )
339 hold on
340 plot (k_grid , k_grid , 'y—' )
341 title ( 'Asset policy function for the good state' )
342 legend ( 'Employed' , 'Unemployed' , '45^0' )
343 xlabel ( 'k today ' )
344 ylabel ( 'k tomorrow' )
345 hold off
346
347 %% Employment distribution
348
349 for t_ind=1:100
350     hist (reshape (N_state (: ,2 ,t_ind) ,1 ,1000) ,40)
351     title ( 'Employment Distribution for 1000 individuals after 100
              periods ' )
352     xticks ([1 2])
353     xticklabels ({ 'Employed' , 'Unemployed' })
354 end
355 print -dpdf histe_fig6.eps
356 %% Compare the asset distribution in equilibrium after 7 periods of
      being in a bad state (low z) as opposed
357 %to being in the high state.
358
359 grouped_b = mat2cell ( i_zb , 1 , diff ( [0 , find (diff (i_zb) ~= 1) ,
      length (i_zb)] )) ;
360
361 for i=1:size (grouped_b ,2)
362     if size (grouped_b {i} ,2)==7
363         g7b = grouped_b {i} ;
364         break
365     end
366 end
367
368 for t=g7b (1) :g7b (7)
369     hist (k_grid (reshape (N_state (: ,1 ,t) ,1 ,1000)) ,40) ;

```



```

370 title( 'Capital Distribution for 1000 individuals after 7 consecutive
        bad periods ')
371 end
372 print -dpdf hist7b_fig7.eps
373
374 W1=(k_grid(reshape(N_state(:,1,t),1,1000)))
375 mean_w1 = mean(W1);
376 disp('Mean of Asset Distribution')
377 disp(mean_w1)
378
379 std_w1 = std(W1);
380 disp('Standard Deviation of Asset Distribution')
381 disp(std_w1)
382
383 skew_w1 =skewness(W1);
384 disp('Skewness of Asset Distribution');
385 disp(skew_w1)
386
387 %% For good
388 grouped_g = mat2cell( i_zg , 1, diff( [0, find(diff(i_zg) ~= 1),
        length(i_zg)] )) ;
389
390 for i=1:size(grouped_g,2)
391     if size(grouped_g{i},2)==7
392         g7g = grouped_g{i};
393         break
394     end
395 end
396
397
398 for t=g7g(1):g7g(7)
399 hist(k_grid(reshape(N_state(:,1,t),1,1000)),40);
400 title( 'Capital Distribution for 1000 individuals after 7 consecutive
        good periods ')
401 end
402 print -dpdf hist7g_fig8.eps
403
404 W2=(k_grid(reshape(N_state(:,1,t),1,1000)));
405 mean_w2 = mean(W1);
406 disp('Mean of Asset Distribution')
407 disp(mean_w2)
408
409 std_w2 = std(W2);
410 disp('Standard Deviation of Asset Distribution')

```

```

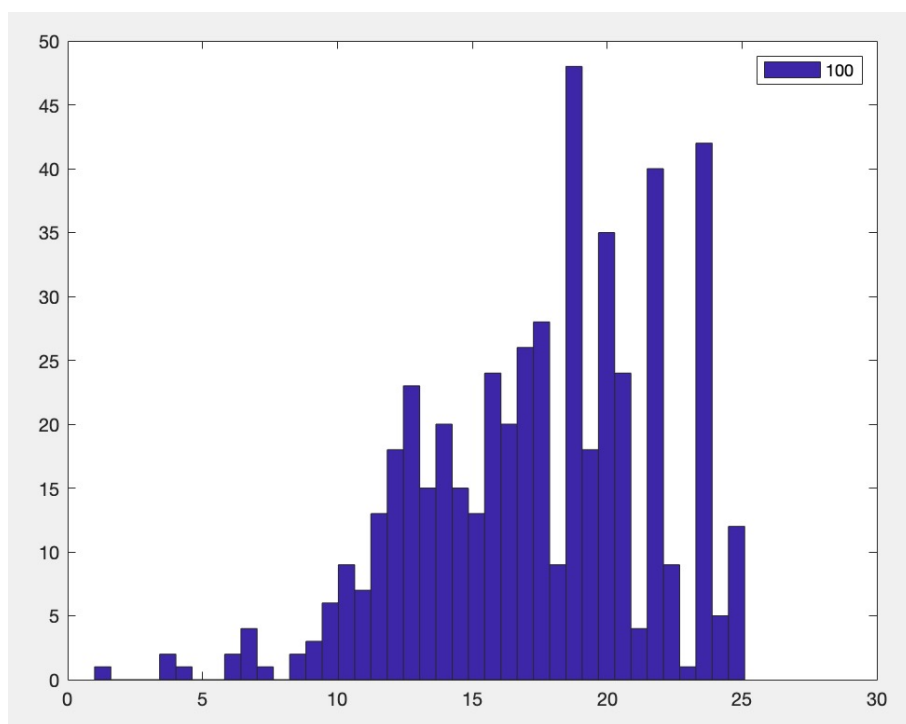
411 disp(std_w2)
412
413 skew_w2 =skewness(W2);
414 disp('Skewness of Asset Distribution');
415 disp(skew_w2)

```

C.3 Heterogeneous expectations.

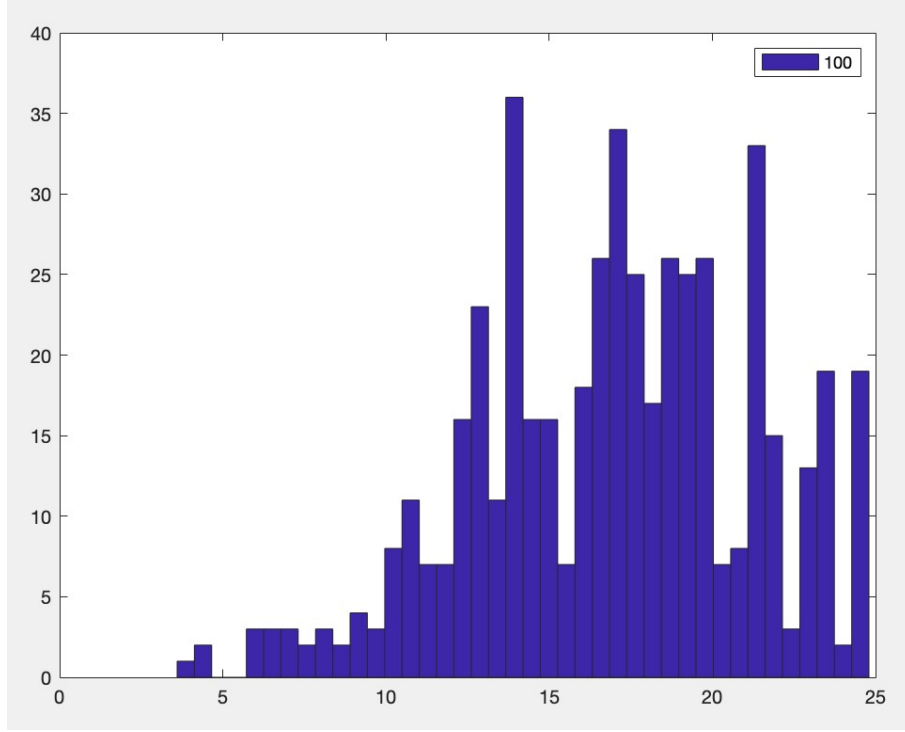
See the Matlab code. We took the assets distribution as an aggregate approximation of the welfare. Thus, figure 6 pictures the distribution for updaters and figure 7 for non-updaters.

Figure 6: Equilibrium asset distribution for updaters



The main differences are that (1) there are more poors between the non-updaters and (2) the average level of wealth is higher for updaters. Thus, the social welfare is going to be lower. Then there are no incentives for a non-updating behavior.

Figure 7: Equilibrium asset distribution for non-updaters



```

1 %% Code for solving a simplified version of K&S (1998)
2 % Quantitative Macroeconomics – IDEA programme
3
4 %% Initial values and parameters
5
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Finding the transition matrix for the state
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8
9 % The system of equations
10 A= [ 1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0 ; ...
11      0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0 ; ...
12      0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0 ; ...
13      0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1 ; ...
14      0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0 ; ...
15      0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0 ; ...
16      0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0 ; ...
17      1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ; ...
18      0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 ; ...
19      0  0  0  0  0  0  0  0  0  5.6 0 -1  0  0  0  0 ; ...
20     -1  0 28/3  0  0  0  0  0  0  0  0  0  0  0  0  0 ; ...
21     .02 .48 .05 .45 0  0  0  0  0  0  0  0  0  0  0  0 ; ...
22      0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 ; ...

```

```

23         0  0  0  0  0  0  0  0  .02  .48  .05  .45  0  0  0  0 ; ...
24         0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 ];
25
26
27 b= [7/8; 7/8; 7/8; 7/8; 1/8; 1/8; 1/8; 1/8; 7/24; 21/40; 0; 0; 0.02;
    0.005; 0.05; 0.02];
28
29
30 pize = reshape(A^-1*b,4,4);
31
32
33
34 % transtion matrix aggregate state
35
36 piZ = [ 7/8  1/8;...
37         1/8  7/8];
38
39
40 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41
42 betta=0.95;
43 delta=0.0025;
44 z=[1.01  0.99];
45 alfa=0.36;
46 L=[0.96,  0.9];
47
48 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Starting values for V %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49
50 v1g = @(k,K) log( alfa*z(1)*(K/L(1))^(alfa-1)*k+ (1-alfa)*z(1)*(K/L
    (1))^(alfa)-delta*k)/(1-betta);
51 v1b = @(k,K) log( alfa*z(2)*(K/L(2))^(alfa-1)*k+ (1-alfa)*z(2)*(K/L
    (2))^(alfa)-delta*k)/(1-betta);
52 v0g = @(k,K) log( alfa*z(1)*(K/L(1))^(alfa-1)*k -delta*k)/(1-betta);
53 v0b = @(k,K) log( alfa*z(2)*(K/L(2))^(alfa-1)*k -delta*k)/(1-betta);
54
55
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Grid for k and K %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57
58 k_grid=[0:0.1:5,5.3:0.3:50];
59 K_grid=[16:0.04:18.5];
60
61 % Evaluation of the VF
62 for j=1:size(K_grid,2)
63 V1g(:,j)= v1g(k_grid,K_grid(j))';

```

```

64 V1b(:,j)= v1b(k_grid,K_grid(j))';
65 V0g(:,j)= v0g(k_grid,K_grid(j))';
66 V0b(:,j)= v0b(k_grid,K_grid(j))';
67 end
68
69
70
71 %%
72 %%%%%%%%% Perceived law of motion %%%%%%%%%
73 % initial values
74
75 b0g=0;
76 b1g=1;
77 b0b=0;
78 b1b=1;
79
80 %%
81 for iter_b=1:1000
82     iter_b
83     % zi is the index for good shock
84     H=@(K,zi) exp( (b0g+b1g*log(K))*zi+ (b0b+b1b*log(K))*(1-zi) );
85
86     % approximation
87
88     Ha= @(K,zi) min(abs(K_grid-H(K,zi)));
89
90
91
92
93 %%% Solution of the consumer problem
94
95
96 % Consumption for each possible decision
97
98 % e=1 employed
99 % g=1 good times =2 bad times
100 c= @(i,I,e,g) max( alfa*z(g)*(K_grid(I)/L(g))^( alfa-1).*k_grid(i)+ ...
101                    (1-alfa)*z(g)*(K_grid(I)/L(g))^( alfa)*e +(1-delta)*
102                    k_grid(i) ...
103                    - k_grid,0) ;
104
105 for iter=1:1000
106     for i=1:size(k_grid,2)

```

```

107     for I=1:size(K_grid,2)
108
109         % approximation next period capital
110
111         [dif,Ip]=min(abs(K_grid-H(K_grid(I),1)));
112         V0gt(i,I)= max(log(c(i,I,0,1))' + betta * ([pize(1,:) ]*([V0g
113             (:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))));
114         V1gt(i,I)= max(log(c(i,I,1,1))' + betta * ([pize(2,:) ]*([V0g
115             (:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))));
116
117         [dif,Ip]=min(abs(K_grid-H(K_grid(I),0)));
118         V0bt(i,I)= max(log(c(i,I,0,2))' + betta * ([pize(3,:) ]*([V0g
119             (:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))));
120         V1bt(i,I)= max(log(c(i,I,1,2))' + betta * ([pize(4,:) ]*([V0g
121             (:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))));
122
123     end
124
125     dev= max(max(abs( [V0gt-V0g,V1gt-V1g,V0bt-V0b,V1bt-V1b] )));
126
127     if dev<0.00001
128         break
129     else
130         V0g=V0gt;
131         V1g=V1gt;
132         V0b=V0bt;
133         V1b=V1bt;
134     end
135
136     end
137
138     % Recover the policy function
139
140     for i=1:size(k_grid,2)
141         for I=1:size(K_grid,2)
142
143             % approximation next period capital
144
145             [dif,Ip]=min(abs(K_grid-H(K_grid(I),1)));
146             [V0gt(i,I),a(i,I,2,1)]= max(log(c(i,I,0,1))' + betta * ([pize

```

```

147     (1,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))');
148     [V1gt(i,I),a(i,I,1,1)]= max(log(c(i,I,1,1))'+ betta * ([pize
149     (2,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))'))');
150
151     [dif,Ip]=min(abs(K_grid-H(K_grid(I),0)));
152     [V0bt(i,I),a(i,I,2,2)]= max(log(c(i,I,0,2))'+ betta * ([pize
153     (3,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))'))');
154     [V1bt(i,I),a(i,I,1,2)]= max(log(c(i,I,1,2))'+ betta * ([pize
155     (4,:) ]*([V0g(:,Ip),V1g(:,Ip),V0b(:,Ip),V1b(:,Ip)]'))'))');
156
157 end
158
159 end
160
161 %%Store the policy functions.
162 assets(:, :, 2, 1) = a(i, I, 2, 1); %employed, bad
163 assets(:, :, 1, 1)= a(i, I, 1, 1); % unemployed, bad
164 assets(:, :, 2, 2) = a(i, I, 2, 2); % employed, good
165 assets(:, :, 1, 2)= a(i, I, 1, 2); %unemployed, good
166
167 if iter==1
168     a_NOupdate = a;
169 end
170 %% Simulation
171
172 % A sequence of TFP
173 % using the index =1 good , =2 bad
174
175 if iter_b==1
176     zt(1)=1;
177
178     for t=2:2000
179         draw=rand;
180         zt(t)= 1+(rand>=piZ(zt(t-1),1));
181     end
182     % Splitting the sample for good and bad times
183
184     % "burning" the first 200 periods
185     ztb=zt;
186     ztb(1:200)=0;
187     % Construct an index for the good times
188     i_zg=find(zt==1);

```

```

187
188 % Construct an index for the bad times
189 i_zb=find(zt==2);
190
191 % initial distribution of assets and employment
192 % =1 employed
193 N_state(1:960,:,1)=ones(960,1)*[26,1];
194 % =2 unemployed
195 N_state(961:1000,:,1)=ones(40,1)*[26,2];
196
197 K_sim(1)=find(K_grid == 17); %The index is for the generated capital
    from the simulated series.
198
199 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Simulating the Distributions
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
200
201 for t=2:2000
202 for n=1:1000
203     if n<500
204 % Evolution of assets
205     N_state(n,1,t)=a_NOupdate(N_state(n,1,t-1),K_sim(t-1),N_state(n
        ,2,t-1),zt(t-1));
206 % Evolution of the employment status
207     N_state(n,2,t)= 2-(rand>=pize(1 + zt(t-1)*2 - N_state(n,2,t-1),zt
        (t)*2-1)/piZ(zt(t-1),zt(t)));
208     else
209 % Evolution of assets
210     N_state(n,1,t)=a_NOupdate(N_state(n,1,t-1),K_sim(t-1),N_state(n
        ,2,t-1),zt(t-1));
211 % Evolution of the employment status
212     N_state(n,2,t)= 2-(rand>=pize(1 + zt(t-1)*2 - N_state(n,2,t-1),zt
        (t)*2-1)/piZ(zt(t-1),zt(t)));
213     end
214
215 end
216
217 % Storage of the sequence of aggregate capital
218 [dev2, K_sim(t)]=min(abs(k_grid(round(mean(N_state(:,1,t)))-K_grid))
    ;
219
220
221 end
222
223 else

```



```

224
225
226 for t=2:2000
227 for n=1:1000
228     if n<=500
229 % Evolution of assets
230     N_state(n,1,t)=a_NOupdate(N_state(n,1,t-1),K_sim(t-1),N_state(n
        ,2,t-1),zt(t-1));
231     N_state_NU(n,1,t)=a_NOupdate(N_state(n,1,t-1),K_sim(t-1),N_state(
        n,2,t-1),zt(t-1));
232     else
233     N_state(n,1,t)=a(N_state(n,1,t-1),K_sim(t-1),N_state(n,2,t-1),zt(
        t-1));
234     end
235 end
236
237 % Storage of the sequence of aggregate capital
238 [dev2, K_sim(t)]=min(abs(k_grid(round(mean(N_state(:,1,t)))-K_grid))
    ;
239
240
241 end
242
243 end
244
245 % Regression model for the evolution of aggregate capital
246
247 % regression for good times (burning the first 20 periods of g times)
248
249 Yg=log(K_grid(K_sim(i_zg(20:end)))');
250 Xg=[ones(size(i_zg(20:end),2),1),log(K_grid(K_sim(i_zg(20:end))-1))')]
    ;
251 Bg=Xg\Yg
252 b0gp=Bg(1);
253 b1gp=Bg(2);
254 % regression for bad times (burning the first 20 periods of bad times
    .
255
256 Yb=log(K_grid(K_sim(i_zb(20:end)))');
257 Xb=[ones(size(i_zb(20:end),2),1),log(K_grid(K_sim(i_zb(20:end))-1))')]
    ;
258 Bb=Xb\Yb
259 b0bp=Bb(1);
260 b1bp=Bb(2);

```

```

261
262
263 dev_b=max(abs([b0g-b0gp b1g-b1gp b0b-b0bp b1b-b1bp]))
264
265 pause(1)
266 if dev_b<=0.01
267     break
268 end
269
270 b0g=0.1*b0gp+0.9*b0g;
271 b1g=0.1*b1gp+0.9*b1g;
272 b0b=0.1*b0bp+0.9*b0b;
273 b1b=0.1*b1bp+0.9*b1b;
274
275
276 end
277
278 %% Number of Iterations
279 disp('Number of Iterations')
280 disp(iter_b)
281
282 %% Mean of the Simulated series
283 mean_s = mean(K_grid(K_sim));
284 disp('Simulated series Aggregate/Average Capital is')
285 disp(mean_s)
286
287 %% R2 for the regression
288
289 mdlg = fitlm(Xg(:,2),Yg);
290 R2g = mdlg.Rsquared.Ordinary;
291 stdg= mdlg.RMSE;
292 disp('R2 for the good shock')
293 disp(R2g)
294
295 mdlb = fitlm(Xb(:,2),Yb);
296 R2b = mdlb.Rsquared.Ordinary;
297 stdb= mdlb.RMSE;
298 disp('R2 for the bad shock')
299 disp(R2b)
300 %% Graphs for the Policy Functions and Asset Distribution
301
302 % Evolution of the Assets distribution
303 %Figure
304 for t_ind=1:100

```

```

305
306     hist(k_grid(reshape(N_state(:,1,t_ind),1,1000)),40)
307     legend(num2str(t_ind))
308     pause(1)
309
310 end
311
312 %% Statistics for the distriburion
313 W = k_grid(reshape(N_state(:,1,t_ind),1,1000));
314
315 mean_w = mean(W);
316 disp('Mean of Asset Distribution')
317 disp(mean_w)
318
319 std_w = std(W);
320 disp('Standard Deviation of Asset Distribution')
321 disp(std_w)
322
323 skew_w=skewness(W);
324 disp('Skewness of Asset Distribution');
325 disp(skew_w)
326
327 %% Retrieve the Policy function from the indices.
328
329 G0g = k_grid(a(:,26,2,1)); % unemployed, good
330 G1g = k_grid(a(:,26,1,1)); % employed, good
331 G0b = k_grid(a(:,26,2,2)); % unemployed, bad
332 G1b = k_grid(a(:,26,1,2)); % employed, bad
333
334 figure(1)
335 plot(k_grid, G1b, 'b' )
336 hold on
337 plot(k_grid, G0b, 'r' )
338 hold on
339 plot(k_grid, k_grid, 'y—')
340 title('Asset policy function for the bad state')
341 legend('Employed', 'Unemployed', '45^0')
342 xlabel('k today ')
343 ylabel('k tomorrow')
344 hold off
345
346 figure(2)
347 plot(k_grid, G1g, 'b' )
348 hold on

```

```

349 plot(k_grid , G0g, 'r' )
350 hold on
351 plot(k_grid ,k_grid , 'y—')
352 title ( 'Asset policy function for the good state' )
353 legend( 'Employed' , 'Unemployed' , '45^0' )
354 xlabel( 'k today ' )
355 ylabel( 'k tomorrow' )
356 hold off
357
358 %% Employment distribution
359
360 for t_ind=1:100
361     hist(reshape(N_state(:,2,t_ind),1,1000),40)
362     title( 'Employment Distribution for 1000 individuals after 100
            periods ' )
363     xticks([1 2])
364     xticklabels({ 'Employed' , 'Unemployed' })
365 end
366 print -dpdf histe_fig6.eps
367 %% Compare the asset distribution in equilibrium after 7 periods of
    being in a bad state (low z) as opposed
368 %to being in the high state.
369
370 grouped_b = mat2cell( i_zb , 1, diff( [0, find(diff(i_zb) ~= 1),
    length(i_zb)] )) ;
371
372 for i=1:size(grouped_b,2)
373     if size(grouped_b{i},2)==7
374         g7b = grouped_b{i};
375         break
376     end
377 end
378
379 for t=g7b(1):g7b(7)
380     hist(k_grid(reshape(N_state(:,1,t),1,1000)),40);
381     title( 'Capital Distribution for 1000 individuals after 7 consecutive
            bad periods ' )
382 end
383 print -dpdf hist7b_fig7.eps
384
385 %% For good
386 grouped_g = mat2cell( i_zg , 1, diff( [0, find(diff(i_zg) ~= 1),
    length(i_zg)] )) ;
387

```

```

388 for i=1:size(grouped_g,2)
389     if size(grouped_g{i},2)==7
390         g7g = grouped_g{i};
391         break
392     end
393 end
394
395
396 for t=g7g(1):g7g(7)
397     hist(k_grid(reshape(N_state(:,1,t),1,1000)),40);
398     title('Capital Distribution for 1000 individuals after 7 consecutive
           good periods ')
399 end
400 print -dpdf hist7g_fig8.eps
401
402 %% Assets for those who do not update
403 G0gNU = k_grid(a_NOupdate(:,26,2,1)); % unemployed, good
404 G1gNU = k_grid(a_NOupdate(:,26,1,1)); % employed, good
405 G0bNU = k_grid(a_NOupdate(:,26,2,2)); % unemployed, bad
406 G1bNU = k_grid(a_NOupdate(:,26,1,2)); % employed, bad
407
408 figure(1)
409 plot(k_grid, G1bNU, 'b' )
410 hold on
411 plot(k_grid, G0bNU, 'r' )
412 hold on
413 plot(k_grid, k_grid, 'y—')
414 title('Asset policy function for the bad state')
415 legend('Employed', 'Unemployed', '45^0')
416 xlabel('k today ')
417 ylabel('k tomorrow')
418 hold off
419
420 figure(2)
421 plot(k_grid, G1gNU, 'b' )
422 hold on
423 plot(k_grid, G0gNU, 'r' )
424 hold on
425 plot(k_grid, k_grid, 'y—')
426 title('Asset policy function for the good state')
427 legend('Employed', 'Unemployed', '45^0')
428 xlabel('k today ')
429 ylabel('k tomorrow')
430 hold off

```

```

431
432 for t_ind=1:100
433
434     hist(k_grid(reshape(N_state(1:500,1,t_ind),1,500)),40)
435     legend(num2str(t_ind))
436     pause(1)
437
438 end
439
440 for t_ind=1:100
441
442     hist(k_grid(reshape(N_state(501:1000,1,t_ind),1,500)),40)
443     legend(num2str(t_ind))
444     pause(1)
445
446 end

```
