ECON 6130 - Problem Set # 5

Julien Manuel Neves

November 2, 2017

Problem 1

1. Sequential problem

$$\begin{aligned} \max & \sum_{t=0}^{\infty} \beta^t u(C_t) \\ \text{such that } & C_t = F(K_{C,t}, L_{C,t}) \\ & I_t = G(K_{I,t}, L_{I,t}) \\ & I_t = I_{C,t} + I_{I,t} \\ & L = L_{C,t} + L_{I,t} \\ & K_{C,t+1} = (1-\delta)K_{I,t} + I_{I,t} \\ & K_{I,t+1} = (1-\delta)K_{C,t} + I_{C,t} \\ & \text{where } L, K_{C,0} \text{ and } K_{I,0} \text{ given.} \end{aligned}$$

Note that we can write $K_{I,t+1}$ in the following way

$$K_{I,t+1} = (1-\delta)(K_{C,t} + K_{I,t}) + G(K_{I,t}, L - L_{C,t}) - K_{C,t+1}$$

Recursive problem

$$\begin{aligned} V(K_{C,t}, K_{I,t}) = \max_{L_{C,t}, K_{C,t+1}} & \{u(F(K_{C,t}, L_{C,t})) \\ & + \beta V(K_{C,t+1}, (1-\delta)(K_{C,t} + K_{I,t}) + G(K_{I,t}, L - L_{C,t}) - K_{C,t+1})\} \end{aligned}$$

where

(i) Control variables.

$$L_{C,t} \text{ and } K_{C,t+1}$$

(ii) State variables.

$$K_{C,t} \text{ and } K_{I,t}$$

2. (1) First-Order conditions.

(i) $K_{C,t+1}$

$$\begin{aligned}
&\beta V_1(K_{C,t+1}, (1-\delta)(K_{C,t}+K_{I,t}) + G(K_{I,t}, L-L_{C,t}) - K_{C,t+1}) \\
-&\beta V_2(K_{C,t+1}, (1-\delta)(K_{C,t}+K_{I,t}) + G(K_{I,t}, L-L_{C,t}) - K_{C,t+1}) = 0 \\
\Rightarrow &V_1(K_{C,t+1}, K_{I,t+1}) = V_2(K_{C,t+1}, K_{I,t+1})
\end{aligned}$$

(ii) $L_{C,t}$

$$\begin{aligned}
&u'(F(K_{C,t}, L_{C,t}))F_2(K_{C,t}, L_{C,t}) \\
-&\beta V_2(K_{C,t+1}, (1-\delta)(K_{C,t}+K_{I,t}) + G(K_{I,t}, L-L_{C,t}) - K_{C,t+1})G_2(K_{I,t}, L-L_{C,t}) = 0 \\
\Rightarrow &u'(C_t)F_2(K_{C,t}, L_{C,t}) = \beta V_2(K_{C,t+1}, K_{I,t+1})G_2(K_{I,t}, L_{I,t})
\end{aligned}$$

(2) Envelope conditions.

Let $X^*_{j,t}$ be the choice of $X_{j,t}$ according to the policy function.

(i) $K_{C,t}$

$$\begin{aligned}
&V_1(K_{C,t}, K_{I,t}) = u'(F(K_{C,t}, L^*_{C,t}))F_1(K_{C,t}, L^*_{C,t}) \\
+&\beta\left\{V_2(K^*_{C,t+1}, (1-\delta)(K_{C,t}+K_{I,t}) + G(K_{I,t}, L-L^*_{C,t}) - K^*_{C,t+1})(1-\delta)\right\} \\
\Rightarrow &V_1(K_{C,t}, K_{I,t}) = u'(C_t{}^*))F_1(K_{C,t}, L^*_{C,t}) + \beta V_2(K^*_{C,t+1}, K^*_{I,t+1})(1-\delta)
\end{aligned}$$

(ii) $K_{I,t}$

$$\begin{aligned}
V_2(K_{C,t}, K_{I,t}) = \beta\big\{&V_2(K^*_{C,t+1}, (1-\delta)(K_{C,t}+K_{I,t}) + G(K_{I,t}, L-L^*_{C,t}) - K^*_{C,t+1}) \\
&\times (1-\delta+G_1(K_{I,t}, L-L^*_{C,t}))\big\} \\
\Rightarrow V_2(K_{C,t}, K_{I,t}) = &\beta V_2(K^*_{C,t+1}, K^*_{I,t+1})(1-\delta+G_1(K_{I,t}, L^*_{I,t}))
\end{aligned}$$

3. Let $X_j = X_{j,t} = X_{j,t+1}$, where $j = \{C, I, \emptyset\}$ and $X = \{K, L, C\}$.

Note that in the steady state we have $X^*_j = X_j$ and

$$\begin{aligned}
K_I + K_C &= \frac{G(K_I, L_I)}{\delta} \\
L_I + L_C &= L
\end{aligned}$$

Then, combining the first-order and envelope conditions, we get

$$\begin{aligned}
V_1(K_C, K_I) &= V_2(K_C, K_I) \\
u'(C)F_2(K_C, L_C) &= \beta V_2(K_C, K_I)G_2(K_I, L_I) \\
V_1(K_C, K_I) &= u'(C))F_1(K_C, L_C) + \beta V_2(K_C, K_I)(1-\delta) \\
V_2(K_C, K_I) &= \beta V_2(K_C, K_I)(1-\delta+G_1(K_I, L_I))
\end{aligned}$$

where $K_I = \frac{G(K_I, L_I)}{\delta} - K_C$ and $L_I = L - L_C$.

This reduce to the following system of equations

$$\frac{1}{\beta} = 1 - \delta + \frac{G_2(K_I, L_I)}{F_2(K_C, L_C)} F_1(K_C, L_C)$$
$$\frac{1}{\beta} = 1 - \delta + G_1(K_I, L_I)$$

where $K_I = \frac{G(K_I, L_I)}{\delta} - K_C$ and $L_I = L - L_C$.

4. We can take the previous equations to derive the following

$$\begin{aligned}
\Rightarrow \frac{G_2(K_I, L_I)}{F_2(K_C, L_C)} F_1(K_C, L_C) &= G_1(K_I, L_I) \\
\frac{G_2(K_I, L_I)}{G_1(K_C, L_C)} &= \frac{F_2(K_I, L_I)}{F_1(K_C, L_C)} \\
\frac{(1-\gamma) K_I^{\gamma} L_I^{-\gamma}}{\gamma K_I^{\gamma-1} L_I^{1-\gamma}} &= \frac{(1-\alpha) K_C^{\alpha} L_C^{-\alpha}}{\alpha K_C^{\alpha-1} L_C^{1-\alpha}} \\
\frac{(1-\gamma)}{\gamma} \frac{K_I}{L_I} &= \frac{(1-\alpha)}{\alpha} \frac{K_C}{L_C}
\end{aligned}$$

Additionally,

$$\begin{aligned}
\frac{1}{\beta} &= 1 - \delta + G_1(K_I, L_I) \\
\frac{1}{\beta} &= 1 - \delta + \gamma K_I^{\gamma-1} L_I^{1-\gamma} \\
\frac{1}{\beta} &= 1 - \delta + \gamma \left(\frac{K_I}{L_I} \right)^{\gamma-1} \\
\Rightarrow \frac{K_I}{L_I} &= \left(\frac{\gamma}{\frac{1}{\beta} - 1 + \delta} \right)^{\frac{1}{1-\gamma}} \\
\Rightarrow \frac{K_C}{L_C} &= \frac{\alpha(1-\gamma)}{\gamma(1-\alpha)} \left(\frac{\gamma}{\frac{1}{\beta} - 1 + \delta} \right)^{\frac{1}{1-\gamma}}
\end{aligned}$$

Then, we can solve for L_C in the following way

$$\begin{aligned}
K_I &= \frac{G(K_I, L_I)}{\delta} - K_C \\
\frac{K_I}{L_I} &= \left(\frac{K_I}{L_I}\right)^{\gamma} \frac{1}{\delta} - \frac{K_C}{L_I} \\
\frac{K_I}{L_I} &= \left(\frac{K_I}{L_I}\right)^{\gamma} \frac{1}{\delta} - \frac{K_C}{L_C}\frac{L_C}{(1-L_C)} \\
\left(\frac{\gamma}{\frac{1}{\beta}-1+\delta}\right)^{\frac{1}{1-\gamma}} &= \left(\frac{\gamma}{\frac{1}{\beta}-1+\delta}\right)^{\frac{\gamma}{1-\gamma}} \frac{1}{\delta} - \frac{\alpha(1-\gamma)}{\gamma(1-\alpha)}\left(\frac{\gamma}{\frac{1}{\beta}-1+\delta}\right)^{\frac{1}{1-\gamma}} \frac{L_C}{(L-L_C)} \\
\frac{L_C}{(L-L_C)} &= \frac{\gamma(1-\alpha)}{\alpha(1-\gamma)}\left(\frac{1}{\delta}\left(\frac{\frac{1}{\beta}-1+\delta}{\gamma}\right)-1\right)
\end{aligned}$$

Let $\Delta = \frac{\gamma(1-\alpha)}{\alpha(1-\gamma)}\left(\frac{1}{\delta}\left(\frac{\frac{1}{\beta}-1+\delta}{\gamma}\right)-1\right)$, then

$$\begin{aligned}
L_C &= \frac{\Delta}{1+\Delta}L \\
L_I &= \frac{1}{1+\Delta}L \\
K_C &= \frac{\alpha(1-\gamma)}{\gamma(1-\alpha)}\left(\frac{\gamma}{\frac{1}{\beta}-1+\delta}\right)^{\frac{1}{1-\gamma}} \frac{\Delta}{1+\Delta}L \\
K_I &= \left(\frac{\gamma}{\frac{1}{\beta}-1+\delta}\right)^{\frac{1}{1-\gamma}} \frac{1}{1+\Delta}L
\end{aligned}$$

Problem 2

1. Recursive problem

$$V(k_t, y_t) = \max_{0 \le k_{t+1} \le e^y k_t^{\alpha} + (1-\delta)k_t} \left\{ u(e^y k_t^{\alpha} + (1-\delta)k_t - k_{t+1}) + \beta \sum_{y_{t+1}} \pi(y_{t+1} \mid y_t) V(k_{t+1}, y_{t+1}) \right\}$$

where k_0, y_0 given

where $\pi(y_{t+1} \mid y_t) = Prob(y = y_{t+1} \mid y = y_t)$ and

(i) Control variables.

$$k_{t+1}$$

(ii) State variables.

$$k_t \text{ and } y_t$$

Note that $0 \leq k_{t+1} \leq e^y k_t^\alpha + (1-\delta)k_t$ is a non-empty, compact, continuous, convex and monotone set.

Thus, for $V(\cdot)$ to be continuous, monotone and concave, we need the following assumptions on $u(\cdot)$:

(a) Continuous and bounded

(b) Strictly increasing in c_t

(c) Strictly concave

2. Note that the code is given in the appendix.

Figure 1 gives the sample distribution of 10000 replications of a 1000 period Markov chain.

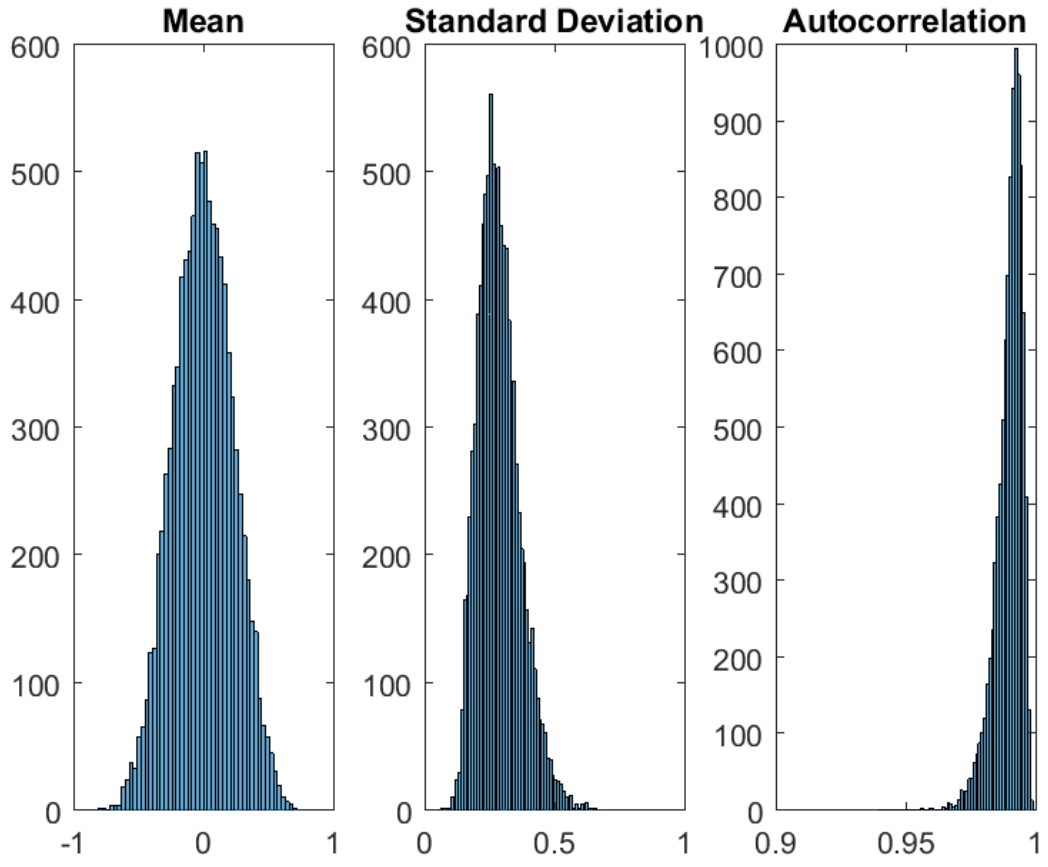

Figure 1: Distribution of Markov Chain

(i) Mean: 0.0003

(ii) Standard Deviation: 0.2867

(iii) Autocorrelation: 0.9898

3. Policy function

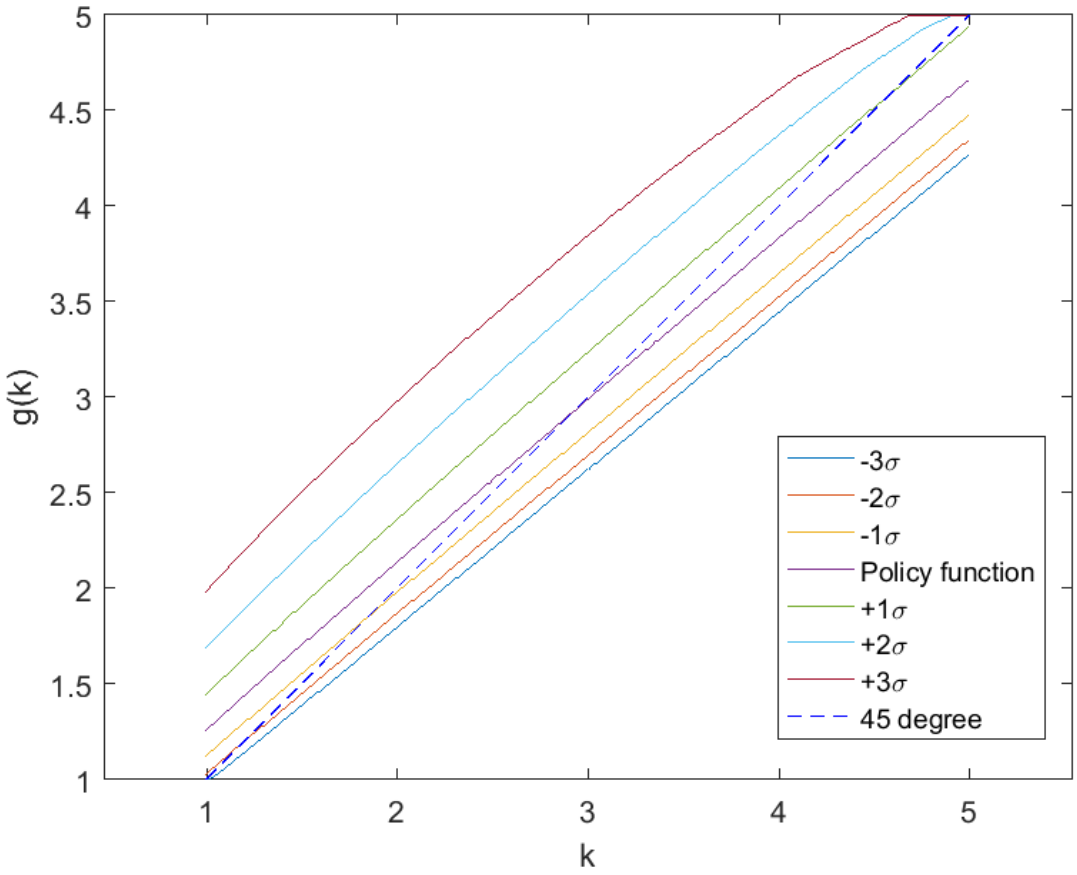

Figure 2: Policy Function

Value function

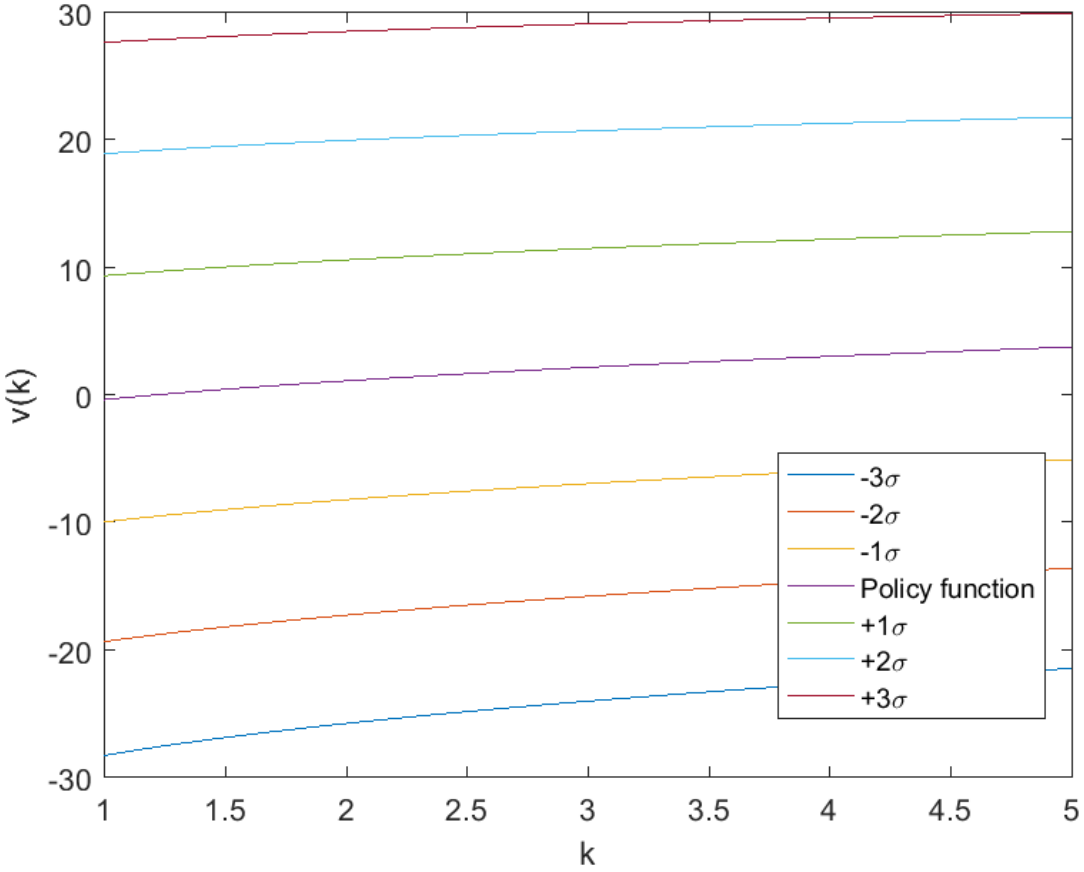

Figure 3: Value Function

4. The simulation is terrible! As we can see in Figure 4, it **does not** match the data at all. I am in pain just looking at the graph. Thankfully, if we change the values of σ_y and ϕ to 0.02 and 0.85 respectively we get Figure 5 which somewhat fits the data.

   The standard deviations are given in Table 1.

	σ_C	σ_{GDP}	σ_I
Data	0.0125	0.0161	0.0739
Simulation	0.2140	0.2168	0.2393

Table 1: Standard Deviations of GDP, Consumption and Investment

The correlations are given in Table 2.

	C	GDP	I
Data	0.0421	0.0674	0.0302

Table 2: Correlations of GDP, Consumption and Investment

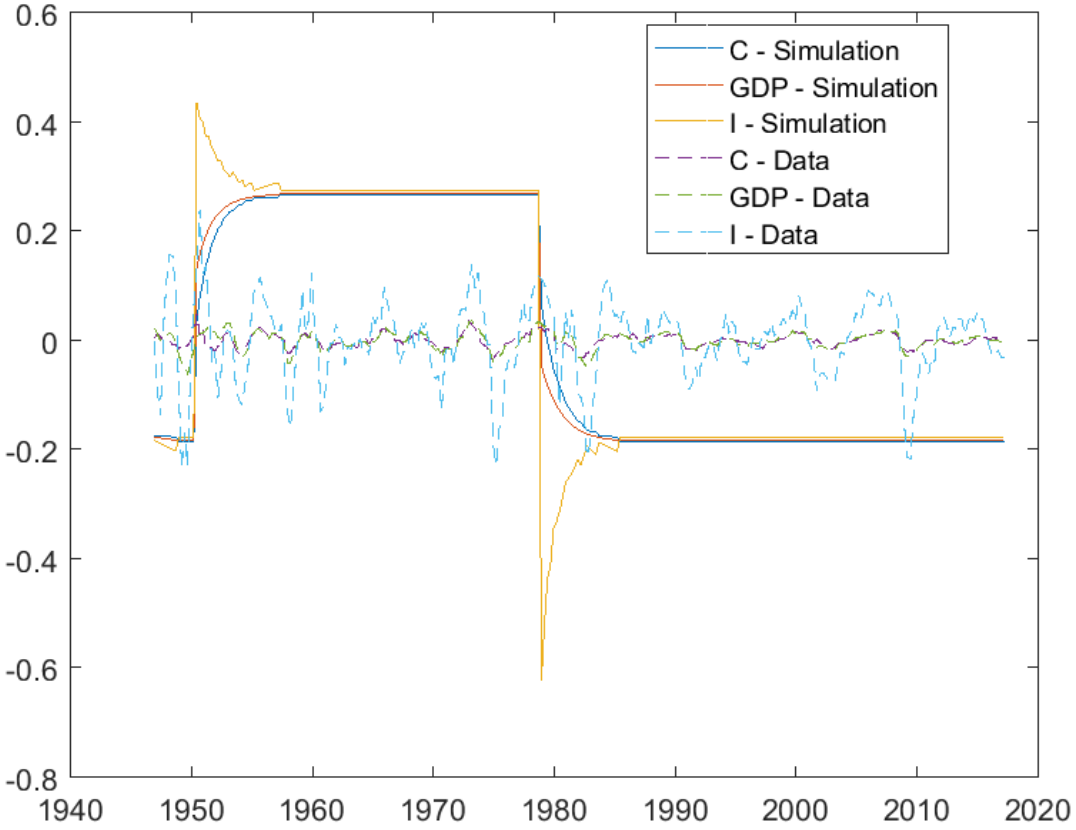

Figure 4: Simulation of RBC Model

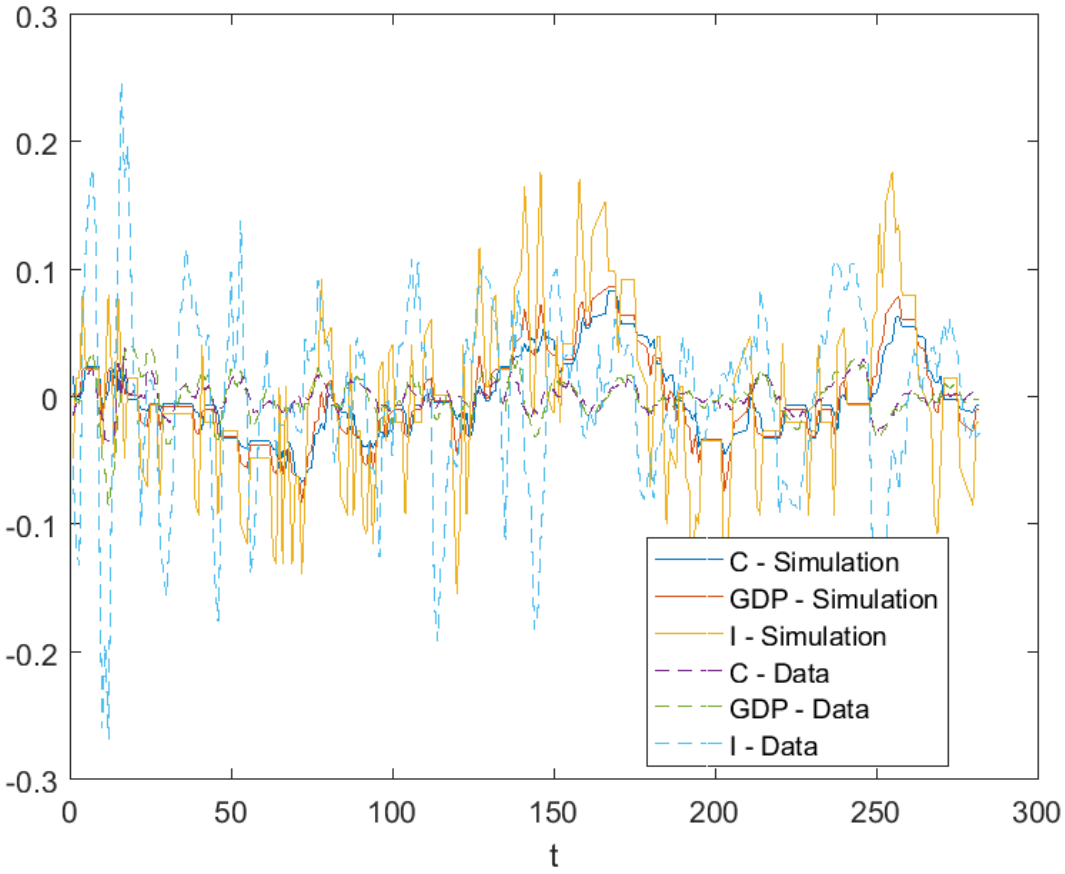

Figure 5: Simulation of RBC Model

Problem 3

1. Sequential Problem

$$\begin{aligned} &\max \sum_{t=0}^{\infty} \beta^t U(c_t, C_t) \\ \text{such that } &C_t = F(K_t, N_t) + (1-\delta)K_t - K_{t+1} \\ &c_t = C_t \\ &K_0 \text{ given.} \end{aligned}$$

Note that the households will maximizes utility when $N_t = 1$, thus we can write $F(K_t, 1) = f(K_t)$.

Recursive Problem

$$V(K_t) = \max_{0 \le K_{t+1} \le f(K_t)+(1-\delta)K_t} \{u(f(K_t) + (1-\delta)K_t - K_{t+1}, f(K_t) + (1-\delta)K_t - K_{t+1}) + \beta V(K_{t+1})\}$$

2. (1) First-Order conditions.

$$-u_1(C_t, C_t) - u_2(C_t, C_t) + \beta V'(K_{t+1}) = 0$$

where $C_t = f(K_t) + (1-\delta)K_t - K_{t+1}$

(2) Envelope conditions.

$$V'(K_t) = u_1(C_t^*, C_t^*)f'(K_t) + u_2(C_t^*, C_t^*)f'(K_t)$$

where $C_t^* = f(K_t) + (1-\delta)K_t - g(K_t)$ and $g(\cdot)$ is the policy function.

Euler equation

$$\begin{aligned}
&u_1(f(K_t) + (1-\delta)K_t - g(K_t), f(K_t) + (1-\delta)K_t - g(K_t)) \\
&+ u_2(f(K_t) + (1-\delta)K_t - g(K_t), f(K_t) + (1-\delta)K_t - g(K_t)) \\
&= \beta \left[u_1(f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t)), f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t)))\right. \\
&\left. + u_2(f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t)), f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t)))\right] f'(g(K_t))
\end{aligned}$$

or

$$u_1(C_t^*, C_t^*) + u_2(C_t^*, C_t^*) = \beta \left[u_1(C_t^{**}, C_t^{**}) + u_2(C_t^{**}, C_t^{**})\right] f'(g(K_t))$$

where $C_t^* = f(K_t) + (1-\delta)K_t - g(K_t)$ and $C_t^{**} = f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t))$.

3. Sequential Problem

$$\begin{aligned}
&\max \sum_{t=0}^{\infty} \beta^t U(c_t, C_t) \\
&\text{such that } c_t = F(K_t, N_t) + (1-\delta)K_t - K_{t+1} \\
&K_0, C_t \text{ given.}
\end{aligned}$$

Note that the households will maximizes utility when $N_t = 1$, thus we can write $F(K_t, 1) = f(K_t)$. Moreover, in this setting the households are taking C_t to be given.

Recursive Problem

$$\begin{aligned}
V(K_t) = &\max_{0 \le K_{t+1} \le f(K_t) + (1-\delta)K_t} \{u(f(K_t) + (1-\delta)K_t - K_{t+1}, C_t) + \beta V(K_{t+1})\} \\
&\text{where } C_t \text{ given}
\end{aligned}$$

4. (1) First-Order conditions.

$$-u_1(f(K_t) + (1-\delta)K_t - K_{t+1}, C_t) + \beta V'(K_{t+1}) = 0$$

where $C_t = f(K_t) + (1-\delta)K_t - K_{t+1}$.

(2) Envelope conditions.

$$V'(K_t) = u_1(f(K_t) + (1-\delta)K_t - g(K_t), C_t^*)f'(K_t)$$

where $C_t^* = f(K_t) + (1-\delta)K_t - g(K_t)$ and $g(\cdot)$ is the policy function.

Euler equation

$$u_1(f(K_t) + (1-\delta)K_t - g(K_t), f(K_t) + (1-\delta)K_t - g(K_t))$$
$$= \beta \left[u_1(f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t)), f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t)))\right] f'(g(K_t))$$

or

$$u_1(C_t^*, C_t^*) = \beta \left[u_1(C_t^{**}, C_t^{**})\right] f'(g(K_t))$$

where $C_t^* = f(K_t) + (1-\delta)K_t - g(K_t)$ and $C_t^{**} = f(g(K_t)) + (1-\delta)g(K_t) - g(g(K_t))$.

5. Note that the social planner solution is different than the unique competitive equilibrium if $u_2(\cdot) \neq 0$.

   Since the social planner has more information than the household (it understands how the household impact the aggregate outcome), the utility generated from the social planner problem is going to be higher than the competitive equilibrium.

   Hence, the competitive equilibrium is not Pareto efficient.

Code

Problem 2

```
%% Value Function Iteration
clear, clc;

%% Set parameters
nk = 500;          % Size of grid
ns = 7;
dev = 3;

delta = 0.1;      % Depreciation rate
alpha = 0.3;      % Capital share of income
beta = 0.96;       % Discount factor

phi = 0.98; % AR coefficient
sig_y = sqrt(.1); % Standard deviation of y_t

%% Markov (Tauchen)
N = 10000;  % Number of simulation
```

```
18
19 sample = ones(N,3); % Initialize mean, std, autocorr. placeholder
20
21 [prob, state_grid] = tauchen(ns, phi, sig_y ,dev); % Transition
      matrix
22 for i = 1:N
23 chain = markovchain(prob, 1000, 4); % Generate Markov chain
24 chain = state_grid(chain); % Map markov chain to state values
25
26 acf = autocorr(chain,1); % Compute autocorrelation
27
28 sample(i,:) = [mean(chain),std(chain), acf(2)]; % Store values
29 end
30
31 mean(sample) % Compute average of mean, std, autocorr.
32
33 %% Value function
34 k_max = 5;        % Upper bound
35 k_min = 1;        % Lower bound
36 k_grid = linspace(k_min,k_max,nk)';      % Create grid
37
38 % Compute value/policy function
39 [val_fun, pol_fun] = bellman( k_grid, state_grid, alpha, beta,
      delta, prob);
40
41 %% Data
42 url = 'https://fred.stlouisfed.org/';
43 c = fred(url);
44
45 GDP = fetch(c,'GDPC1'); % Fetch GDP from FRED
46 CON = fetch(c,'PCECC96');    % Fetch consumption from FRED
47 INV = fetch(c,'GPDIC1');     % Fetch investment from FRED
48
49 gdp_data = log(GDP.Data(:,2));  % Log of GDP
50 c_data = log(CON.Data(:,2));     % Log of consumption
51 i_data = log(INV.Data(:,2));     % Log of investment
52
53 [~,gdp_data] = hpfilter(gdp_data,1600); % Extract cyclical
      component of GDP
54 [~,i_data] = hpfilter(i_data,1600); % Extract cyclical component
      of C
55 [~,c_data] = hpfilter(c_data,1600); % Extract cyclical component
      of I
56
57 %% Simulation
```

```
58 T = size(c_data,1)+1;   % Set size of Markov Chain to match data
59
60 y_sim = markovchain(prob, T, 4);    % Generate Markov chain
61
62 k_sim = ones(T,1); % Initialize capital vector
63 k_sim(1) = round(nk/2); % Set starting value to the middle of grid
64
65 for t = 2:T
66     k_sim(t)= pol_fun(k_sim(t-1),y_sim(t-1));   % Compute k' from
       k
67 end
68
69 y_sim = state_grid(y_sim); % Map markov chain to state values
70 k_sim = k_grid(k_sim); % Map markov chain to capital values
71
72 gdp_sim = exp(y_sim).* k_sim.^alpha;    % Compute GDP
73 gdp_sim = gdp_sim(1:end-1); % Drop last value
74
75 i_sim= k_sim(2:end) -(1-delta)*k_sim(1:end-1); % Compute
     investment
76
77 c_sim = gdp_sim - i_sim;    % Compute consumption
78
79 c_sim = log(c_sim); % Log of c
80 i_sim = log(i_sim); % Log of i
81 gdp_sim = log(gdp_sim); % Log of GDP
82
83 c_sim = c_sim-mean(c_sim); % Detrend c
84 i_sim = i_sim- mean(i_sim); % Detrend i
85 gdp_sim = gdp_sim - mean(gdp_sim); % Detrend GDP
86
87 std_sim = std([c_sim,gdp_sim,i_sim]) % Compute standard value of
     simulation
88 std_data = std([c_data,gdp_data,i_data]) % Compute standard value
     of data
89 corr(c_sim,c_data) % Compute the correlation for consumption
90 corr(gdp_sim,gdp_data) % Compute the correlation for GDP
91 corr(i_sim,i_data) % Compute the correlation for investment
92
93 %% Plot figures
94 % Value function
95 figure
96 plot(k_grid,val_fun)
97
98 xlabel('k')
```

```
ylabel('v(k)')
legend('-3\sigma','-2\sigma','-1\sigma','Policy function','+1\
    sigma','+2\sigma','+3\sigma', 'Location', 'best')

print('plot_value','-dpng')

% Policy function
figure
plot(k_grid,k_grid(pol_fun))
axis equal
hold on
plot(k_grid,k_grid, '--b')

xlabel('k')
ylabel('g(k)')
legend('-3\sigma','-2\sigma','-1\sigma','Policy function','+1\
    sigma','+2\sigma','+3\sigma','45 degree', 'Location', 'best')

print('plot_policy','-dpng')

% Markov
figure
subplot(1,3,1); histogram(sample(:,1)); title('Mean');
subplot(1,3,2); histogram(sample(:,2)); title('Standard Deviation'
    );
subplot(1,3,3); histogram(sample(:,3)); title('Autocorrelation');

print('plot_markov','-dpng')

% Simulation
figure
plot(GDP.Data(:,1),[c_sim,gdp_sim,i_sim])
hold on
plot(GDP.Data(:,1),[c_data,gdp_data,i_data], '--')

datetick('x')
legend('C - Simulation', 'GDP - Simulation','I - Simulation','C -
    Data','GDP - Data','I - Data', 'Location', 'best')
print('plot_sim','-dpng')
```

Transition Matrix (tauchen.m)

```
function [prob, state_grid] = tauchen(ns, phi, sig_y ,dev)
%markovchain Discretize AR(1) process
%   [prob, cum_prob, state_grid] = tauchen(ns, phi, sig_y ,dev)
```

```
    returns
%   prob, transition matrix,
%   cum_prob, cummulative distribution
%   state_grid, value of states

state_grid = linspace(-dev*sig_y,dev*sig_y,ns)';    % Set state
    grid
% Compute y^j_t+1 - phi y^i_t
state_change = repmat(state_grid,1,ns)' - phi*repmat(state_grid
    ,1,ns);

sig_eps = sig_y*sqrt(1-phi^2);  % Compute std of epsilon

prob = zeros(ns,ns);    % Initialize transition matrix
% Compute transition matrix according to Tauchen (1986)
prob(:,1) = cdf('norm',state_change(:,1)+sig_y/2,0,sig_eps);
prob(:,end) = 1- cdf('norm',state_change(:,end) - sig_y/2,0,
    sig_eps);
prob(:,2:end-1) = cdf('norm',state_change(:,2:end-1)+sig_y/2,0,
    sig_eps)-cdf('norm',state_change(:,2:end-1)-sig_y/2,0,sig_eps);

end
```

Markov Chain (markovchain.m)

```
function [chain] = markovchain(prob, T, start)
%markovchain Generate Markov chain
%   [chain] = markovchain(prob, T, start) returns chain, Markov
    chain.

chain = ones(T,1);  % Initialize Markov Chain
chain(1)= start;    % Set starting value

cum_prob = cumsum(prob,2);  % Compute cumulative distribution

% Generate Markov Chain using random numbers uniformly distributed
for t = 2:T
    chain(t)=find(cum_prob(chain(t-1),:)>rand(),1);
end

end
```

Value Function Iteration (bellman.m)

```

```

```
function [val_fun, pol_fun] = bellman( k_grid, state_grid, alpha,
    beta, delta, prob)
% bellman Value Function Iteration with log utility
%   [val_fun, pol_fun] = bellman( k_grid, state_grid, alpha, beta,
    delta, prob) returns
%   val_fun, value function and pol_fun, policy function

crit = 1;       % Initialize convergence criterion
tol = 1e-6;     % Convergence tolerance

%% Grid
nk = size(k_grid,1);
ns = size(state_grid,1);
% Create grid

% Empty
val_temp = zeros(nk,ns);  % Initialize temporary value function
    vector
val_fun = zeros(nk,ns); % Initialize value function vector
pol_fun = zeros(nk,ns); % Initialize policy function vector


%% Value function iteration
while crit>tol ;
    % Iterate on k
    for j = 1:ns
    for i=1:nk
        c = exp(state_grid(j))* k_grid(i)^alpha + (1-delta)*k_grid
            (i) - k_grid; % Compute consumption for kt
        utility_c = log(c); % Compute utility for every ct
        utility_c(c<=0) = -Inf; % Set utility to -Inf for c<=0
        [val_fun(i,j),pol_fun(i,j)] = max(utility_c + beta*
            val_temp*prob(j,:)');   % Solve Bellman equation
    end
    end
    crit = max(abs(val_fun-val_temp));  % Compute convergence
        criterion
    val_temp = val_fun; % Update value function
end

end
```