Table of Contents


```
% Tianli Xia
% Macro Homework 5
% Dynamic programming: value-iteration
% Value Function:
```

$$V(k,y) = max_{k'} log(e^{y_t}k^{\alpha} - (1-\delta)k - k') + \beta EV(k',y')$$

$EV(k',y') = \Pi(y_{t+1}|y_t)V(k_{t+1}, y_{y+1})$ Policy Function:

$$Q(k,y;k') = log(e^{y_t}k^{\alpha} - (1-\delta)k - k') + \beta EQ(k',y';k'')$$

Other important notations:

- Transition: k=k', P matrix
- State variable: k, y
- Action varible: k'

```
clear all
```

Initialize the parameters

```
alpha=0.35; % the production rate: $f(k)=k^{\alpha}$
beta=0.95; % the intertemporary patience
delta=0.1; % capital deprecation rate
k=100; % number of grids
len= 0.2; % length of grid, it is good to start from big value and
 then decrease.
% In pratice, I set length to be 0.01 at the beginning, at find that
 the
% optimal value at steady state should be within (0, 0.2), hence I set
 the
% length to a lower value to cover this interval.
start= len;
state= start:len:start+len*(k-1); % different states in grids:
action= start:len:start+len*(k-1);
```

Initialize the transiontion matrix P

```
m= 7; % number of discrete points we approximate
lamda= 0.98; % coefficient of AR(1) process
sigmaY= sqrt(0.1); % standard deviation of Y
sigmaE= sqrt(1-lamda^2)*sigmaY; % standard deviation of Y
Y(m+2)=inf;
Y(1)= -inf; % Set the boundary value
P(m,m)= 0; % Define (P(t,t-1))
for i=1:m
    Y(i+1)=(i-((m+1)/2))*sigmaY;
end
for i=1:m % Note that in the loop i=1-> Y(i)=-inf, so we need
 P(1,.)=Y(2)
    for j=1:m
        P(i,j)=normcdf(((Y(j+1)+Y(j+2))/2-lamda*Y(i+1))/sigmaE)-...
               normcdf(((Y(j+1)+Y(j))/2-lamda*Y(i+1))/sigmaE);
    end
end
```

Initialize value function matrix

```
V = ones(k,m); % V: state* action matrix
PI = ones(k,m); % PI: best policy matrix
threshold= 10^(-4); % Tolerance level in the loop
epsilon=1; % random initial value of the gao between two loops
count=0; % Count how many times the loop runs

tic
while epsilon> threshold % loop until $\epsilon$ converges
    V_temp=-inf*ones(k,m,k); % Any infeasible capital brings -inf
 value
    for s=1:k % value function iteration: get current value
        for j=1:m
            a_max=exp(Y(j+1))*state(s)^alpha+ (1-delta)*state(s); %
 Calculate feasible action sets: $0<=a<=state(s)^alpha+ (1-
delta)*state(s)$
            for a=1:min(k, ceil( (a_max-start)/len) )
                V_temp(s,j,a)= log( exp(Y(j+1))*state(s)^alpha+ (1-
delta)*state(s) -action(a)) + beta*P(j,:)*V(a,:)';
                % This directly comes from Bellman Optimality equation
            end
        end
        [V_new, PI]= max(V_temp, [], 3); % Calculate (1) New value
 function; (2) best action function.
    end
    epsilon= ( max(max( abs(V_new -V)))); % Calculate the current
 error
    V=V_new;
    count=count+1; % Count times the loop ends
end
toc
```

```
fprintf("The loop ends in %d runs, the gap within final two loops are
 %f.", count, epsilon)
save("solution.mat")
```

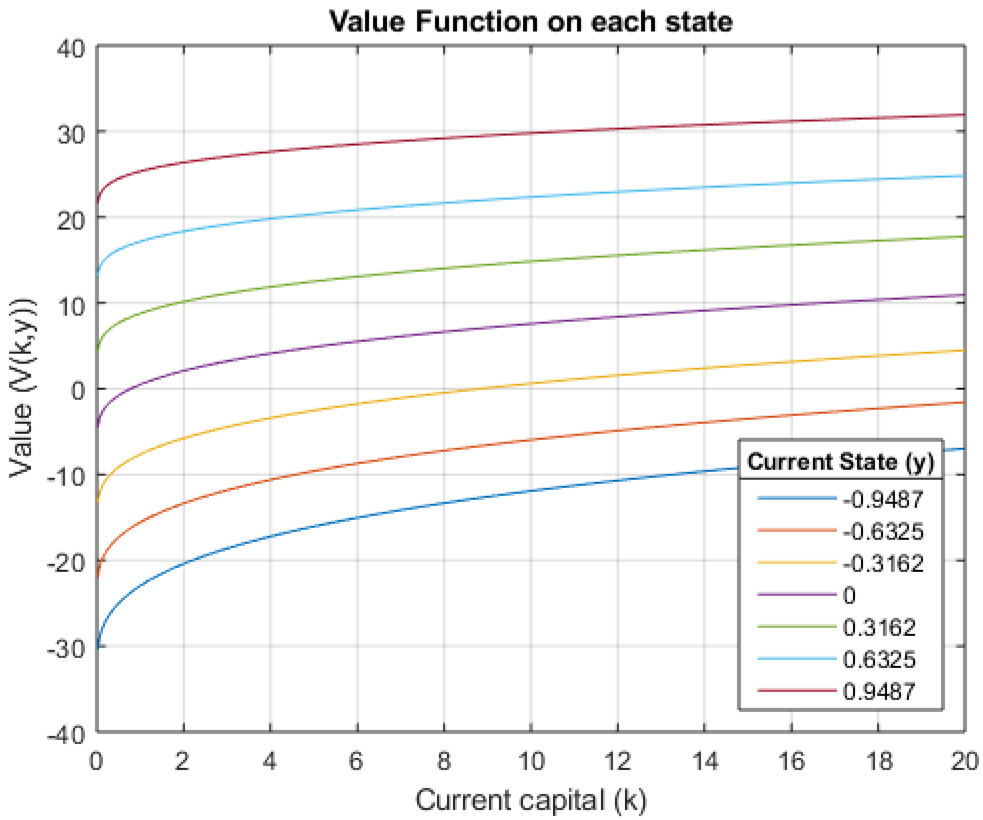
```
Elapsed time is 35.223267 seconds.
The loop ends in 175 runs, the gap within final two loops are
 0.000099.
```

Plot graph

Plot the value function

$V(k,y) = \max_{k'} log(exp^y s^\alpha - (1-\delta)s - k') + \beta P(y'|y)V(k',y')$ I further divide the loop into 1000 grids of capital to get better graphs.

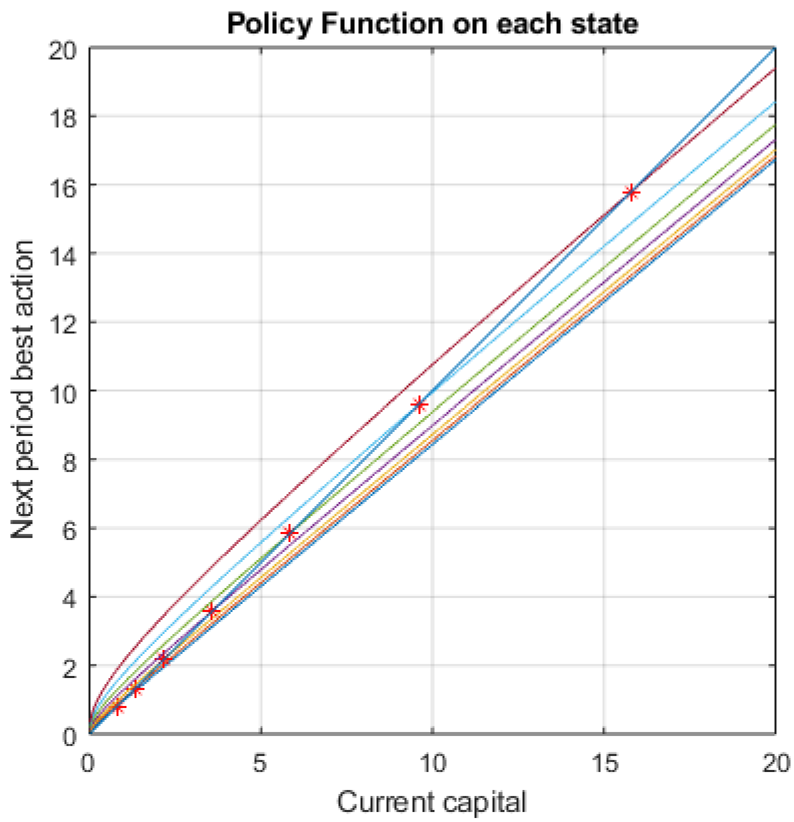
```
if exist('solution_1000.mat', 'file')
    load 'solution_1000.mat'
end

for i=1:7
    plot(state, V(:,i))
    hold on
end
    grid on
    axis on
    xlabel("Current capital (k)")
    ylabel("Value (V(k,y))")
    lgd=
 legend("-0.9487","-0.6325","-0.3162","0","0.3162","0.6325","0.9487",'Location','s
    title(lgd, "Current State (y)")
    title("Value Function on each state")
    hold off

% print -djpeg -r600 hw5_value
```



Plot action function

```
%This plots show the optimal capital decision. At any point on the
 left side
%of steady state, we increase the capital and vice versa. At steady
 state
%it is stable. Hence the steady state is the crosspoint of the policy
%function and the 45 degree line.


for j=1:m
    i(j)=1;
    while i(j)~=PI(i(j),j) % By definition, this is the optimal
 decision
        i(j)=i(j)+1;
    end

    %fprintf("The optimal capital at the steady state is %f\n. The
 value at optimal capital is %f\n" ...
    %    , i*len, V(i,j) )
    %fprintf("If feasible, the planner will always choose steady state
 capital in the next period.")
        plot(state, action(PI(:,j)))
        x=state;
        y=state;
```

```
        hold on
        plot(x(i(j)),y(i(j)),'r*')

    %   print -djpeg -r600 hw5_action
end
    plot(x,y)
    hold on
    grid on
    axis on
    xlabel("Current capital")
    ylabel("Next period best action")
    title("Policy Function on each state")
    axis square
    hold off
```



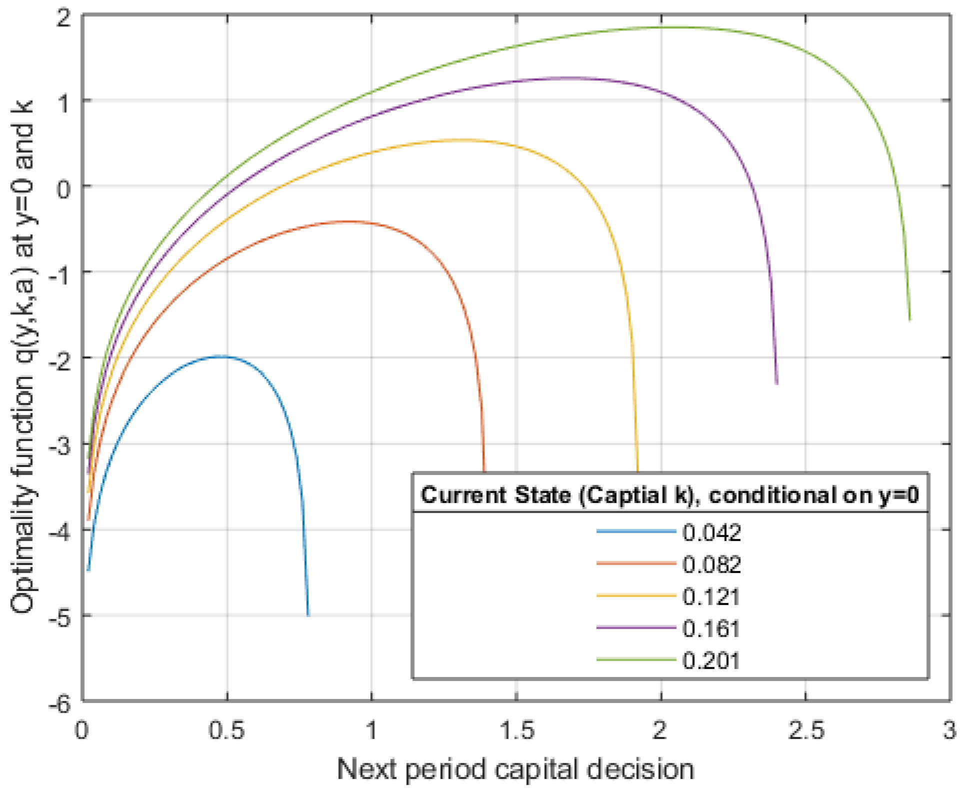
Plot potential function of actions in terms of action at each state

$Q(s, y, a) = \max_k log(exp^y s^\alpha - (1 - \delta)s - k) + \beta V(k)$

```
for i=11:20:91
    y= reshape(V_temp(i,4,:), size(state));
    plot(state, y)
    hold on
end
```

```
xlabel("Next period capital decision")
ylabel("Optimality function q(y,k,a) at y=0 and k")
lgd=
 legend("0.042","0.082","0.121","0.161","0.201","0.241",'Location','southeast');
title(lgd, "Current State (Captial k), conditional on y=0")
axis on
grid on
hold off
```

Warning: Ignoring extra legend entries.



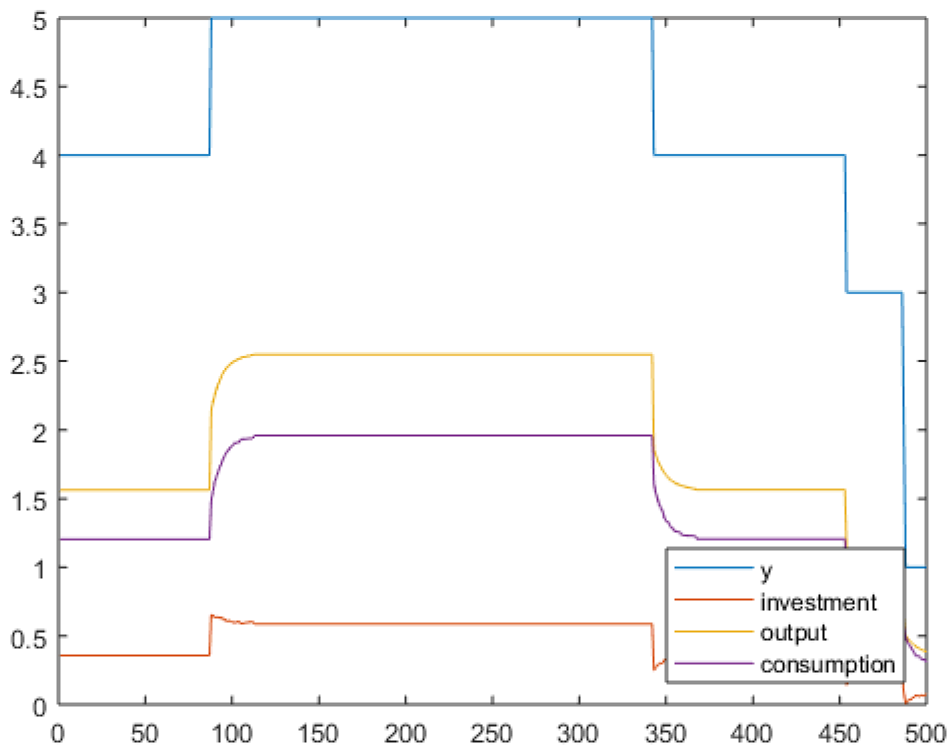
Calcualte the model predicted output, capital, and investment

Start from a steady state, suppose all the decision is optimal (according to PI matrix) Simulate the process for 100 times

```
% method1: simply use the pointwise decision rule and iterate on it
steady=repmat(1:1:k, m, 1);
index= ceil ((sum((steady.*(steady==PI')), 2)./
 sum(((steady==PI')),2 )));  % By definition, this is the optimal
 decision
% Here index denotes the steady state, we set it as the original value
nrep=500;
r=100;
```

```
y=[];
consumption=[];
y(1)=4; % state variable y
start=index(y(1));
capital(nrep)=0;
output(nrep)=0;
investment(nrep)=0;
capital(1)=state(start); % state variable capital
output(1)= exp(Y(y(1)+1))*capital(1)^alpha; % output according to
 production function
investment(1)=  action(PI(start,y(1)))- (1-delta)*capital(1); %
 investment decision
consumption(1)= output(1)-investment(1);

for i=2:nrep
    y(i)=sum( rand(1)>=( cumsum(P(y(i-1),:))))+1; % random process of
 state
    capital(i)= action(PI(action==capital(i-1),y(i-1))); % capital
 accumulation process
    output(i)= exp(Y(y(i)+1))*capital(i)^alpha;
    investment(i)=  action(PI(action==capital(i),y(i)))- (1-
delta)*capital(i);
    consumption(i)= output(i)-investment(i);
end

plot(1:i, y)
hold on
plot(1:i, investment)
hold on
plot(1:i, output)
hold on
plot(1:i, consumption)
hold on
lgd=
 legend('y','investment','output','consumption','Location','southeast');
hold off
fprintf('The variance-covariance matrix of output, investment,
 consumption.')
V1= cov([log(output)' log(investment)' log(consumption)'])

fprintf('The correlation coefficients of output, investment,
 consumption.')
V2= corr([log(output)' log(investment)' log(consumption)'])

The variance-covariance matrix of output, investment, consumption.
V1 =

    0.1444    0.1802    0.1377
    0.1802    0.2478    0.1688
    0.1377    0.1688    0.1319

The correlation coefficients of output, investment, consumption.
V2 =
```

```
1.0000    0.9526    0.9980
0.9526    1.0000    0.9334
0.9980    0.9334    1.0000
```



Alternative method: interpolation scheme to make y continuous

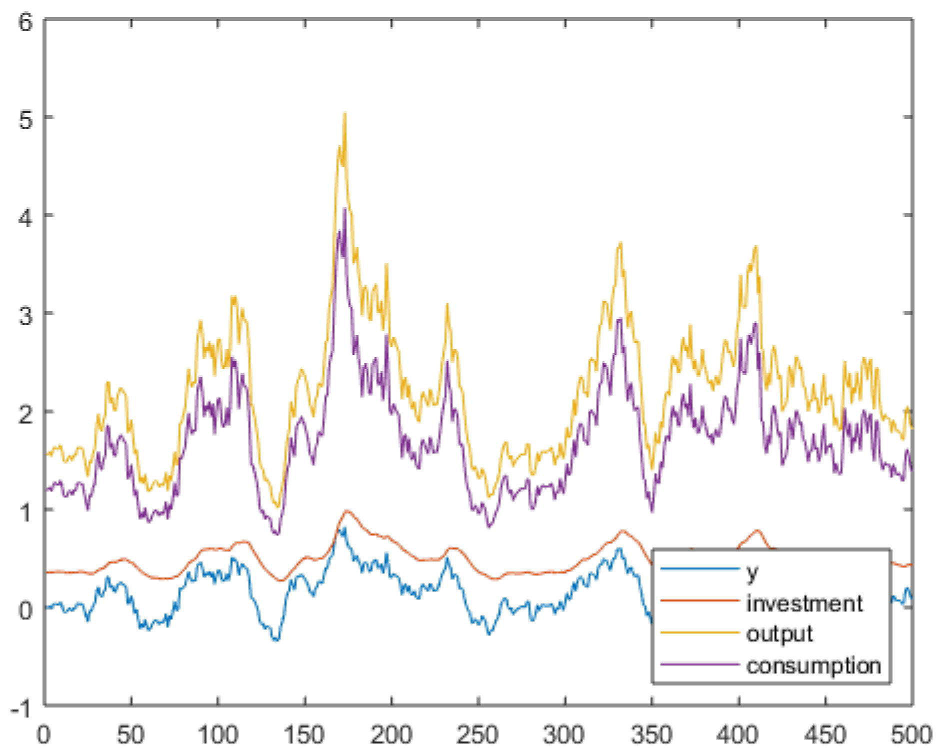
The following method is to interpolate the decision process at any y value. We use the Chebyshev polynomial.

```
if exist('solution_1000.mat', 'file')
    load 'solution_1000.mat'
end
n=7;
steady=repmat(1:1:k, m, 1);
index= ceil ((sum((steady.*(steady==PI')), 2)./
 sum(((steady==PI')),2 )));

nrep=500;
ymax=max(Y(2:8));
ymin=min(Y(2:8));
ygrid= Y(2:8);
transy= 2*(ygrid- ymin)/(ymax- ymin)-1 ;
Ty= [ones(m,1) transy'];
```

```
for i=3:n
    Ty=[Ty 2*transy'.*Ty(:,i-1)-Ty(:,i-2)];
end
for i=1:a
    b(:,i)=(Ty)\PI(i,:)';
end

y(1)=0; % state variable y
start=index(find(Y==y(1))-1);
capital(nrep)=0;
output(nrep)=0;
investment(nrep)=0;
capital(1)=state(start); % state variable capital
output(1)= exp(y(1))*capital(1)^alpha; % output according to
 production function
investment(1)=  action(PI(start,find(Y==y(1))-1))- (1-
delta)*capital(1); % investment decision
consumption(1)= output(1)-investment(1);

for i=2:nrep
    y(i)=0.98*y(i-1)+sigmaE*randn(1); % random process of state
    tryt= 2*(y(i)- ymin)/(ymax- ymin)-1 ;
    Ty= [1 tryt];
    for j=3:n
        Ty=[Ty 2*tryt.*Ty(:,j-1)-Ty(:,j-2)];
    end
    next= round( Ty*b(:,capital(i-1)==state));

    capital(i)= action(next); % capital accumulation process
    output(i)= exp(y(i))*capital(i)^alpha;
    investment(i)=  action(next)- (1-delta)*capital(i);
    consumption(i)= output(i)-investment(i);
end

% Have a look at the time path
plot(1:nrep, y)
hold on
plot(1:nrep, investment)
hold on
plot(1:nrep, output)
hold on
plot(1:nrep, consumption)
hold on
lgd=
 legend('y','investment','output','consumption','Location','southeast');
hold off

% Calucalte the variance-covariance matrix
fprintf('The variance-covariance matrix of output, investment,
 consumption.')
V3= cov([log(output)' log(investment)' log(consumption)'])

% Isaac suggests using the ratio instead of absolute value to avoid
 the
```

```
% problem of unit:
% Calucalte the variance-covariance matrix
fprintf('The correlation coefficients of output, investment,
 consumption.')
V4= corr([log(output)' log(investment)' log(consumption)'])
```

```
The variance-covariance matrix of output, investment, consumption.
V3 =

    0.0902    0.0779    0.0940
    0.0779    0.0774    0.0781
    0.0940    0.0781    0.0989

The correlation coefficients of output, investment, consumption.
V4 =

    1.0000    0.9333    0.9950
    0.9333    1.0000    0.8932
    0.9950    0.8932    1.0000
```

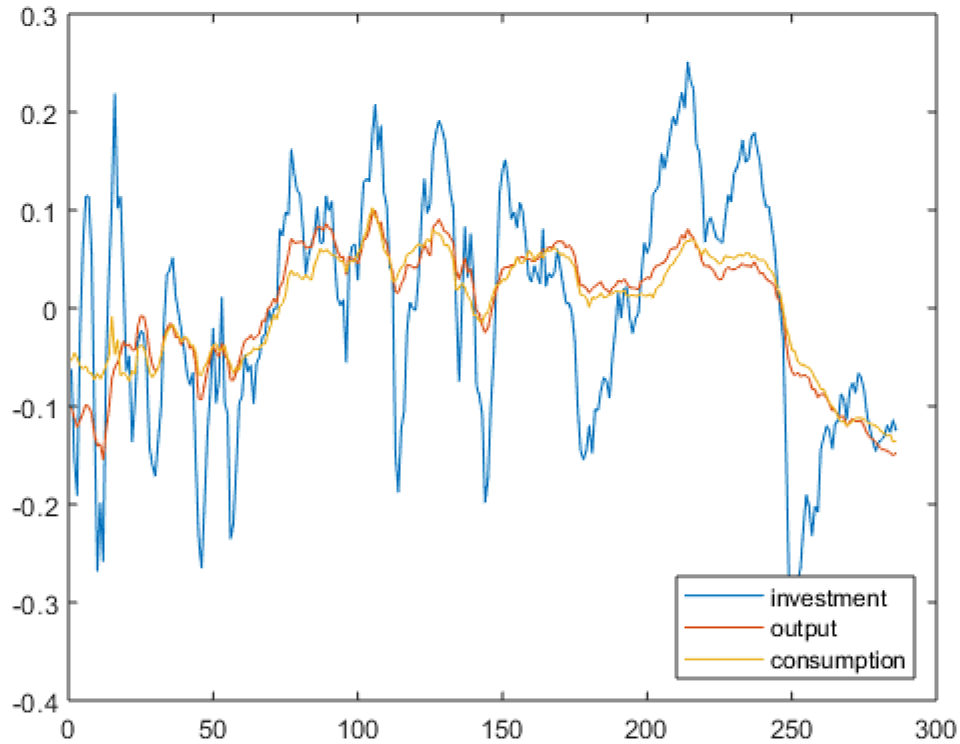


Real data

Read the real data from csv file

```
rgdp=csvread('gdp.csv',1,1);
```

```
rinv=csvread('inv.csv',1,1);
rcon=csvread('consumption.csv',1,1);
detrend_rgdp = detrend(log(rgdp));
detrend_rinv = detrend(log(rinv));
detrend_rcon = detrend(log(rcon));

plot(1:size(rgdp),detrend_rinv)
hold on
plot(1:size(rgdp),detrend_rgdp)
hold on
plot(1:size(rgdp),detrend_rcon)
lgd=
 legend('investment','output','consumption','Location','southeast');
hold off
fprintf('The variance-covariance matrix of output, investment,
 consumption.')
V5= cov([(detrend_rgdp) (detrend_rinv) (detrend_rcon)])
fprintf('The correlation coefficients of output, investment,
 consumption.')
V6= corr([(detrend_rgdp) (detrend_rinv) (detrend_rcon)])

% We found (1) that the correlations in both simulation and real data
 are very high,
% which makes sense because they are highly related. Consumption is
 more
% associated with output than investment; (2) the correlation
% model predicts is even higher, probably because in the model there
 is no
% noise; (3) the standard deviation, because of the problem of units,
 the
% variance seems different from model prediction. But the ratio of
% deviations in all the model are pretty close.

The variance-covariance matrix of output, investment, consumption.
V5 =

    0.0044    0.0058    0.0037
    0.0058    0.0149    0.0050
    0.0037    0.0050    0.0034

The correlation coefficients of output, investment, consumption.
V6 =

    1.0000    0.7185    0.9561
    0.7185    1.0000    0.7044
    0.9561    0.7044    1.0000
```



Published with MATLAB® R2018a