Table of Contents


```
% Tianli Xia
% Macro Homework 3
% Dynamic programming: value-iteration
% Policy Function:
```

$$Q(s,k) = log(s^{\alpha} - (1-\delta)s - k) + \beta V(k)$$

```
% Value Function:
```

$$V(s) = max_k Q(s,k)$$

Other important notations:

- Transition: s=k
- State variable: s
- Action varible: k

```
clear all
```

Initialize the parameters

```
alpha=0.3; % the production rate: $f(k)=k^{\alpha}$
beta=0.6; % the intertemporary patience
delta=0.75; % capital deprecation rate
k=200; % number of grids
len= 0.001; % 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 value function matrix

```
V = ones(k,1); % V: state* action matrix
```

```
PI = ones(k,1); % PI: best policy matrix
threshold= 10^(-6); % 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

while epsilon> threshold % loop until $\epsilon$ converges
    V_temp=-inf*ones(k); % Any infeasible capital brings -inf value
    for s=1:k % value function iteration: get current value
        a_max=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,a)= log( state(s)^alpha+ (1-delta)*state(s) -
action(a)) + beta* V(a);
            % This directly comes from Bellman Optimality equation
        end
        [V_new, PI]= max(V_temp, [], 2); % Calculate (1) New value
 function; (2) best action function.
    end
    epsilon= ( max( abs(V_new -V))); % Calculate the current error
    V=V_new;
    count=count+1; % Count times the loop ends
end

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

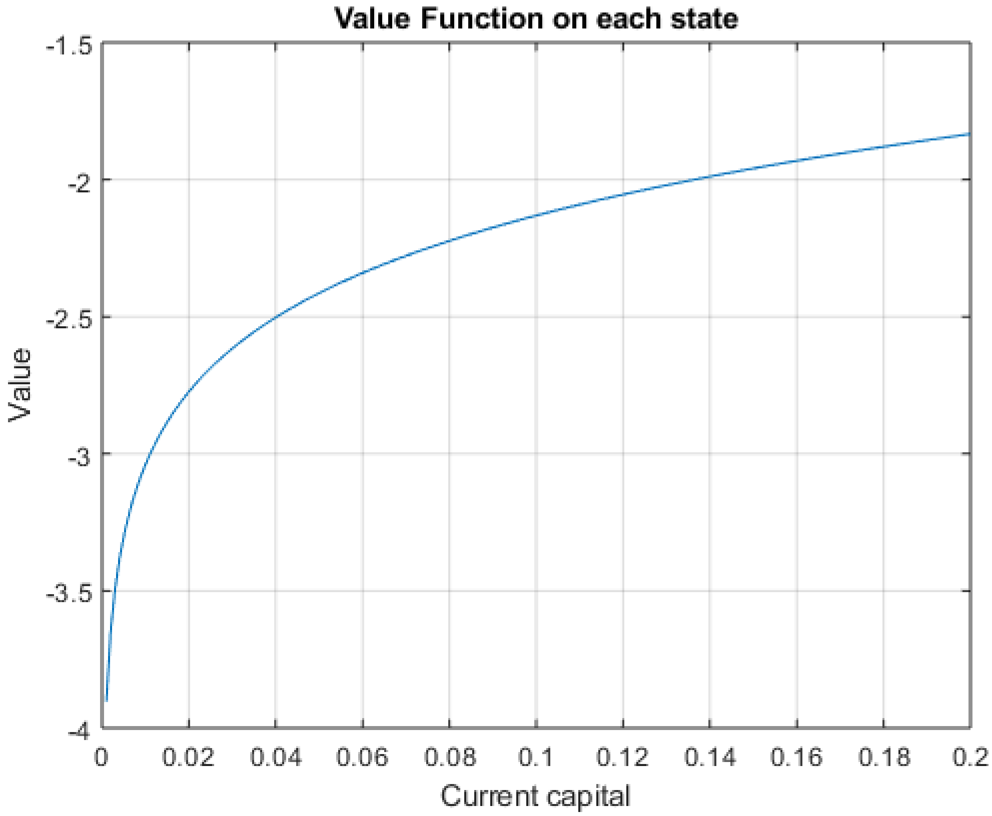
The loop ends in 29 runs, the gap within final two loops are 0.000001.

Plot graph

Plot the value function

% V(k)= \max_{k} log(s^{\alpha} - (1-\delta)s - k) + \beta V(k)$

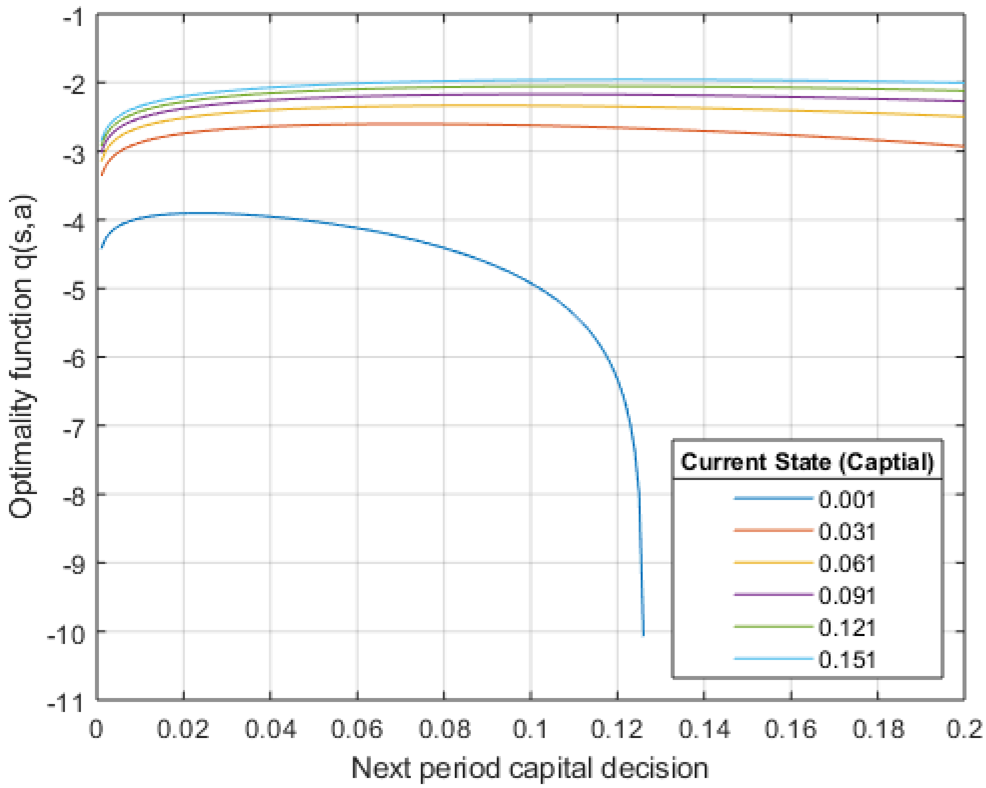
```
plot(state, V)
grid on
axis on
xlabel("Current capital")
ylabel("Value")
title("Value Function on each state")

% print -djpeg -r600 hw3_value_2
```



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

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

```
for i=1:30:151
    plot(state, V_temp(i,:))
    hold on
end
xlabel("Next period capital decision")
ylabel("Optimality function q(s,a)")
lgd=
 legend("0.001","0.031","0.061","0.091","0.121","0.151",'Location','southeast');
title(lgd, "Current State (Captial)")
axis on
grid on
hold off
% for i=1:5:26
%     plot(state(PI(i)), V_temp(i,PI(i)), "r.")
%     hold on
% end
% hold off
%print -djpeg -r600 -hw3_control_2
```

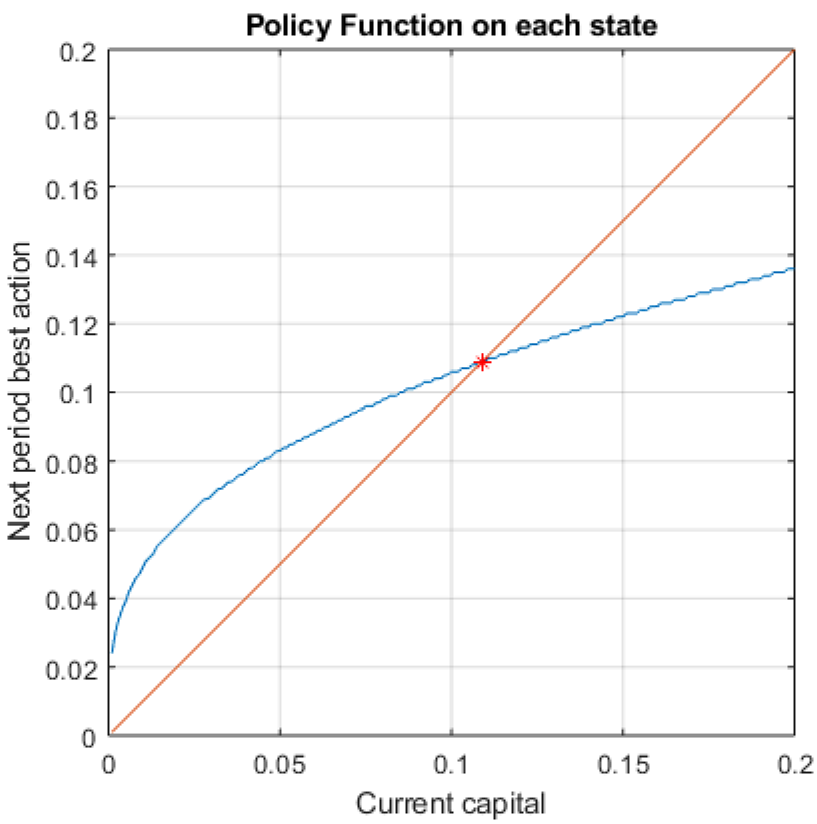


Plot action function

```
%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.
i=1;
while i~=PI(i) % By definition, this is the optimal decision
    i=i+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) )
fprintf("If feasible, the planner will always choose steady state
 capital in the next period.")
    plot(state, action(PI))
    x=state;
    y=state;
    hold on
    plot(x,y)
    hold on
    plot(x(i),y(i),'r*')
```

```
    grid on
    axis on
    xlabel("Current capital")
    ylabel("Next period best action")
    title("Policy Function on each state")
    axis square
%   print -djpeg -r600 hw3_action_2
```

```
The optimal capital at the steady state is 0.109000
. The value at optimal capital is -2.095064
If feasible, the planner will always choose steady state capital in
 the next period.
```



Published with MATLAB® R2018a