ps3_caiy

January 30, 2019

Exercise 5.1.

Since the individual only lives for one period and u' >0, the objective function is

$$\max_{W_{t+1}\in[0,W_t]} u(W_t - W_{t+1}) \tag{1}$$

i.e.

$$\max_{W_{t+1}\in[0,W_t]} W_t - W_{t+1} \tag{2}$$

Therefore, the condition that characterizes the optimal amount of cake to eat in period 1 is: W_2 = $W_{t+1} = 0$

Exercise 5.2.

Since the individual lives for 2 period and u' >0, to optimize the amount of cake to leave for the next period W_3 in period 2 is to:

$$\max_{W_3\in[0,W_2]} u(W_2 - W_3) \tag{3}$$

that is: $W_3 = 0$, i.e. c_2 = W_2

To optimize the amount of cake leave for the next period W_2 in period 1 is to:

$$\max_{W_2\in[0,W_1]} u(W_1 - W_2) + \beta * u(W_2) \tag{4}$$

That is:

$$u'(W_1 - W_2) = \beta * u'(W_2) \tag{5}$$

Exercise 5.3.

(1)If the individual lives for three periods T = 3:

To opitimize W_4:

$$\max_{W_4\in[0,W_3]} u(W_3 - W_4) \tag{6}$$

that is: $W_4 = 0$, i.e. c_3 = W_3

To opitimize W_3:

$$\max_{W_3\in[0,W_2]} u(W_2 - W_3) + \beta * u(W_3) \tag{7}$$

F.O.C:

$$u'(W_2 - W_3) = \beta * u'(W_3) \tag{8}$$

define

$$\psi(W_t) \equiv W_{t+1} \tag{9}$$

which satisfies:

$$u'(W_t - W_{t+1}) = \beta * u'(W_{t+1} - W_{t+2}) \tag{10}$$

Then:

$$W_3 = \psi(W_2) \tag{11}$$

To opitimize W_2:

$$\max_{W_2 \in [0, W_1]} u(W_1 - W_2) + \beta * u(W_2 - \psi(W_2)) + \beta^2 * u(\psi(W_2)) \tag{12}$$

F.O.C:

$$u'(W_1 - W_2) = \beta * u'(W_2 - \psi(W_2)) \tag{13}$$

i.e.:

$$W_2 = \psi(W_1) \tag{14}$$

(2)Assume:

$$W_1 = 1 \tag{15}$$
$$\beta = 0.9 \tag{16}$$
$$u(c_t) = \ln(c_t) \tag{17}$$

According to above equations:

$$\begin{cases} \frac{1}{(W_2 - W_3)} = \frac{0.9}{(W_3 - 0)} \\ \frac{1}{(1 - W_2)} = \frac{0.9}{(W_2 - W_3)} \end{cases} \tag{18}$$

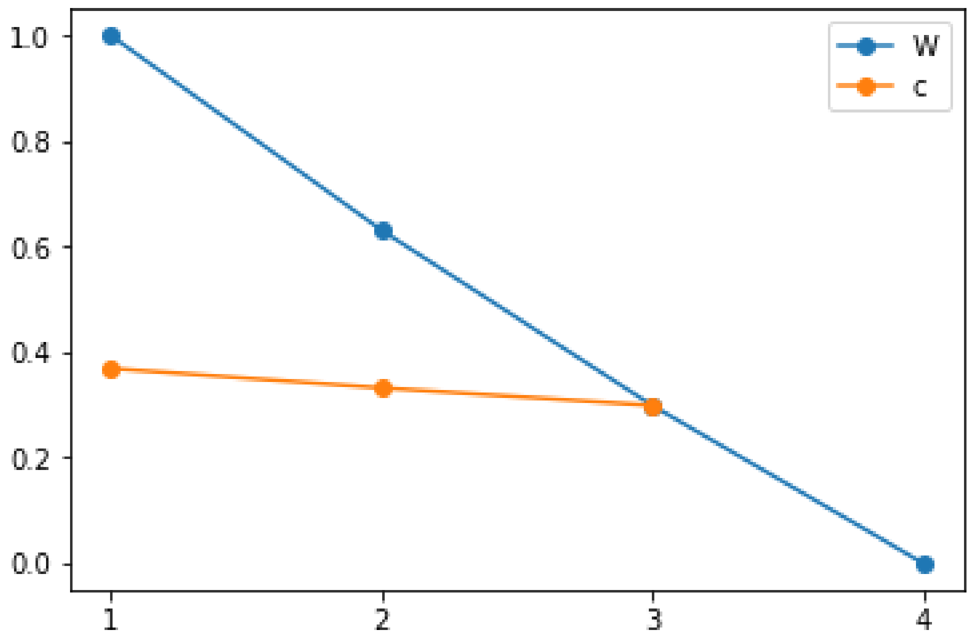
Thus:

$$W_1 = 1 \tag{19}$$
$$W_2 = 0.6310 \tag{20}$$
$$W_3 = 0.2989 \tag{21}$$
$$W_4 = 0 \tag{22}$$

which means:

$$c_1 = 0.3690 \tag{23}$$
$$c_2 = 0.3321 \tag{24}$$
$$c_3 = 0.2989 \tag{25}$$

```python
In [50]: import matplotlib.pyplot as plt
         W = [1, 0.6310, 0.2989, 0]
         c = [0.3690,0.3321,0.2989]
         T1 = [1,2,3,4]
         T2 = [1,2,3]
         %matplotlib inline
         plt.plot(T1,W,marker='o',label="W")
         plt.plot(T2,c,marker='o',label="c")
         plt.legend()
         plt.xticks([1,2,3,4])
         plt.show()
```



Exercise 5.4.
By definition:

$$V_{T-1}(W_{T-1}) = \max_{W_T, W_{T+1}} u(W_{T-1} - W_T) + \beta * u(W_T - W_{T+1}) \tag{26}$$

As in utility maximization situation:

$$W_{T+1} = \psi(W_T) = 0 W_T = \psi(W_{T-1}) \tag{27}$$

The value function $V_{T-1}(W_{T-1})$ will be:

$$V_{T-1}(W_{T-1}) = \max_{W_T} u(W_{T-1} - W_T) + \beta * u(W_T) = u(W_{T-1} - \psi(W_{T-1})) + \beta * u(\psi(W_{T-1})) \tag{28}$$

Exercise 5.5.

When T < ∞ and $u(c)$ = $\ln(c)$:

$$u'(W_{T-1} - W_T) = \beta * u'(W_T - 0) \tag{29}$$

which gives:

$$\psi_{T-1}(\bar{W}) = \frac{\bar{W}}{\beta + 1} \tag{30}$$

However:

$$\psi_T(\bar{W}) = \bar{W} = 0 \tag{31}$$

Thus:

$$\psi_{T-1}(\bar{W}) \neq \psi_T(\bar{W}) \tag{32}$$

For value function:

$$V_{T-1}(\bar{W}) = \ln(\bar{W} - \frac{(\bar{W})}{\beta + 1}) + \beta * \ln(\frac{(\bar{W})}{\beta + 1}) V_T(\bar{W}) = \ln(\bar{W}) \tag{33}$$

if $V_{T-1}(\bar{W})$ = $V_T(\bar{W})$:

$$\ln(\beta) + \ln(\bar{W}) - \ln(\beta + 1) + \beta * \ln(\bar{W}) - \beta * \ln(\beta + 1) = \ln(\bar{W}) \tag{34}$$

which means:

$$\beta * \ln(\bar{W}) = (\beta + 1) * \ln(\beta + 1) > 0 \tag{35}$$

i.e.:

$$\bar{W} > 1 \tag{36}$$

Thus:

$$V_{T-1}(\bar{W}) \neq V_T(\bar{W}) \tag{37}$$

Exercise 5.6.
u(c) = ln(c), write the finite horizon Bellman equation for the value function at time T-2:

$$V_{T-2}(W_{T-2}) = \max_{W_{T-1}, W_T, W_{T+1}} u(W_{T-2} - W_{T-1}) + \beta * u(W_{T-1} - W_T) + \beta^2 * (W_T - W_{T+1}) \tag{38}$$

Using envolope theroem:

$$\begin{cases} \frac{1}{(W_{T-2} - W_{T-1})} = \frac{\beta}{(W_{T-1} - W_T)} \\ \frac{1}{(W_{T-1} - W_T)} = \frac{\beta}{(W_T)} \end{cases} \tag{39}$$

Solution:

$$\begin{cases} W_{T-1} = \frac{\beta^2 + \beta}{\beta^2 + \beta + 1} * W_{T-2} \\ W_T = \frac{\beta}{\beta + 1} * W_{T-1} = \frac{\beta^2}{\beta^2 + \beta + 1} * W_{T-2} \end{cases} \tag{40}$$

Then:

$$V_{T-2}(W_{T-2}) = \ln\left(\frac{W_{T-2}}{\beta^2+\beta+1}\right) + \beta \times \ln\left(\frac{\beta \times W_{T-2}}{\beta^2+\beta+1}\right) + \beta^2 \times \ln\left(\frac{\beta^2 \times W_{T-2}}{\beta^2+\beta+1}\right) \tag{41}$$

Exercise 5.7.
analytical solutions for $\psi_{T-s}(W_{T-s})$ and $V_{T-s}(W_{T-s})$ for the general integer $s \geq 1$:

$$\psi_{T-s}(W_{T-s}) = \psi_{T-s}(\psi_{T-s-1}(W_{T-s-1}) = \psi^{T-s}(W_1) = \frac{\sum_{i=1}^{s}\beta^i}{1+\sum_{i=1}^{s}\beta^i} W_{T-s} V_{T-s}(W_{T-s}) = \sum_{i=1}^{s}\beta^{s-i} * u(W_{T-i} - \psi(W_{T-i} \tag{42}$$

When $s \to \infty$:

$$\psi(W_{T-s}) = \beta W_{T-s} V(W_{T-s}) = \left(\frac{1}{1-\beta}\right)\ln((1-\beta)W_{T-s}) + \frac{\beta}{(1-\beta)^2}\ln(\beta) \tag{43}$$

Exercise 5.8.

$$V(W) = \max_{W' \in [0,W]} u(W-W')) + \beta V(W') \tag{44}$$

Exercise 5.9.

```
In [11]: #Approximate the continuum of possible cake sizes by a column vector
         import numpy as np
         W_vec = np.linspace(0.01,1,100)
         print(W_vec)
```

```
[0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1  0.11 0.12 0.13 0.14
 0.15 0.16 0.17 0.18 0.19 0.2  0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28
 0.29 0.3  0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4  0.41 0.42
 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5  0.51 0.52 0.53 0.54 0.55 0.56
 0.57 0.58 0.59 0.6  0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.7
 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8  0.81 0.82 0.83 0.84
 0.85 0.86 0.87 0.88 0.89 0.9  0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98
 0.99 1.  ]
```

Exercise 5.10.
the resulting policy function $W' = \psi(W)$ is deicided by:

$$V_T(W) = \max_{W' \in [0.01,1]} ln(W-W') + 0.9\ln(W') \tag{45}$$

Because $W_{min} = 0.01$, Then: $W' = \psi(W) = 0.01$

```
In [37]: # find V_T(W)
         def utility(x):
             ut = np.log(x)
             return ut
```

```python
beta = 0.9

c_mat = np.tile(W_vec.reshape(100,1),(1,100))-np.tile(W_vec.reshape(1,100)
c_pos = c_mat >0
c_mat[~c_pos] = 1e-7
u_mat = utility(c_mat)

v_tp1 = np.zeros(100)
v_prime = np.tile(v_tp1.reshape(1,100),(100,1))
v_prime[~c_pos]= -9e+4 #punish the upper diagonal entries ,not to choose t
v_t = (u_mat + beta*v_prime).max(axis = 1)

print("V_T(W) is",v_t)

#confirm our analytical solution of psi(W)
index=np.argmax(u_mat+beta*v_prime,axis=1)
W_prime = W[index]
print("W'= psi(W)=",W_prime)
```

```
V_T(W) is [-8.10161181e+04 -4.60517019e+00 -3.91202301e+00 -3.50655790e+00
 -3.21887582e+00 -2.99573227e+00 -2.81341072e+00 -2.65926004e+00
 -2.52572864e+00 -2.40794561e+00 -2.30258509e+00 -2.20727491e+00
 -2.12026354e+00 -2.04022083e+00 -1.96611286e+00 -1.89711998e+00
 -1.83258146e+00 -1.77195684e+00 -1.71479843e+00 -1.66073121e+00
 -1.60943791e+00 -1.56064775e+00 -1.51412773e+00 -1.46967597e+00
 -1.42711636e+00 -1.38629436e+00 -1.34707365e+00 -1.30933332e+00
 -1.27296568e+00 -1.23787436e+00 -1.20397280e+00 -1.17118298e+00
 -1.13943428e+00 -1.10866262e+00 -1.07880966e+00 -1.04982212e+00
 -1.02165125e+00 -9.94252273e-01 -9.67584026e-01 -9.41608540e-01
 -9.16290732e-01 -8.91598119e-01 -8.67500568e-01 -8.43970070e-01
 -8.20980552e-01 -7.98507696e-01 -7.76528789e-01 -7.55022584e-01
 -7.33969175e-01 -7.13349888e-01 -6.93147181e-01 -6.73344553e-01
 -6.53926467e-01 -6.34878272e-01 -6.16186139e-01 -5.97837001e-01
 -5.79818495e-01 -5.62118918e-01 -5.44727175e-01 -5.27632742e-01
 -5.10825624e-01 -4.94296322e-01 -4.78035801e-01 -4.62035460e-01
 -4.46287103e-01 -4.30782916e-01 -4.15515444e-01 -4.00477567e-01
 -3.85662481e-01 -3.71063681e-01 -3.56674944e-01 -3.42490309e-01
 -3.28504067e-01 -3.14710745e-01 -3.01105093e-01 -2.87682072e-01
 -2.74436846e-01 -2.61364764e-01 -2.48461359e-01 -2.35722334e-01
 -2.23143551e-01 -2.10721031e-01 -1.98450939e-01 -1.86329578e-01
 -1.74353387e-01 -1.62518929e-01 -1.50822890e-01 -1.39262067e-01
 -1.27833372e-01 -1.16533816e-01 -1.05360516e-01 -9.43106795e-02
 -8.33816089e-02 -7.25706928e-02 -6.18754037e-02 -5.12932944e-02
 -4.08219945e-02 -3.04592075e-02 -2.02027073e-02 -1.00503359e-02]
W'= psi(W)= [0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
```

```
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
 0.01 0.01]
```

Exercise 5.11.

```
In [23]: def dist(v1,v2):
             dist=sum((v1-v2)**2)
             return dist
```

Exercise 5.12.

```
In [36]: beta = 0.9

         c_mat = np.tile(W_vec.reshape(100,1),(1,100))-np.tile(W_vec.reshape(1,100)
         c_pos = c_mat >0
         c_mat[~c_pos] = 1e-7
         u_mat = utility(c_mat)


         v_prime = np.tile(v_t.reshape(1,100),(100,1))
         v_prime[~c_pos]= -9e+4 #punish the upper diagonal entries ,not to choose t
         v_tm1 = (u_mat + beta*v_prime).max(axis = 1)

         print("V_{T-1}(W) is",v_tm1)

         #confirm our analytical solution of psi(W)
         index=np.argmax(u_mat+beta*v_prime,axis=1)
         W_prime = W[index]
         print("W'= psi-{T-1}(W)=",W_prime)
```

```
V_{T-1}(W) is [-8.10161181e+04 -7.29191115e+04 -8.74982335e+00 -8.05667617e+00
 -7.43284371e+00 -7.02737860e+00 -6.66246000e+00 -6.37477793e+00
 -6.11586407e+00 -5.89272052e+00 -5.69189132e+00 -5.50956976e+00
 -5.34548036e+00 -5.19132968e+00 -5.05259407e+00 -4.91906268e+00
 -4.79888442e+00 -4.68110139e+00 -4.57509666e+00 -4.46973614e+00
 -4.37442596e+00 -4.27960150e+00 -4.19259012e+00 -4.10681096e+00
 -4.02676825e+00 -3.94845801e+00 -3.87435004e+00 -3.80231160e+00
 -3.73331873e+00 -3.66662156e+00 -3.60208303e+00 -3.53998945e+00
 -3.47936483e+00 -3.42128016e+00 -3.36412175e+00 -3.30955959e+00
 -3.25549236e+00 -3.20404979e+00 -3.15275650e+00 -3.10396633e+00
 -3.05530583e+00 -3.00878582e+00 -2.96262185e+00 -2.91817009e+00
 -2.87425894e+00 -2.83169933e+00 -2.78983132e+00 -2.74900932e+00
 -2.70900273e+00 -2.66978202e+00 -2.63147837e+00 -2.59373804e+00
 -2.55699824e+00 -2.52063060e+00 -2.48533196e+00 -2.45024064e+00
 -2.41627434e+00 -2.38237279e+00 -2.34958297e+00 -2.31685209e+00
```

```
 -2.28510339e+00 -2.25352120e+00 -2.22274954e+00 -2.19223815e+00
 -2.16238519e+00 -2.13287434e+00 -2.10388681e+00 -2.07531298e+00
 -2.04714210e+00 -2.01944761e+00 -1.99204864e+00 -1.96518097e+00
 -1.93851272e+00 -1.91242394e+00 -1.88644845e+00 -1.86109466e+00
 -1.83577685e+00 -1.81108424e+00 -1.78642517e+00 -1.76232761e+00
 -1.73832619e+00 -1.71479569e+00 -1.69141776e+00 -1.66842824e+00
 -1.64564221e+00 -1.62316935e+00 -1.60094600e+00 -1.57896710e+00
 -1.55727930e+00 -1.53577310e+00 -1.51459565e+00 -1.49354224e+00
 -1.47285167e+00 -1.45223238e+00 -1.43200681e+00 -1.41180411e+00
 -1.39200148e+00 -1.37222046e+00 -1.35280238e+00 -1.33344679e+00]
W'= psi(W)= [0.01 0.01 0.02 0.02 0.03 0.03 0.04 0.04 0.05 0.05 0.06 0.06 0.07 0.07
 0.08 0.08 0.09 0.09 0.1  0.1  0.1  0.11 0.11 0.12 0.12 0.13 0.13 0.14
 0.14 0.15 0.15 0.16 0.16 0.17 0.17 0.18 0.18 0.19 0.19 0.19 0.2  0.2
 0.21 0.21 0.22 0.22 0.23 0.23 0.24 0.24 0.25 0.25 0.26 0.26 0.27 0.27
 0.28 0.28 0.28 0.29 0.29 0.3  0.3  0.31 0.31 0.32 0.32 0.33 0.33 0.34
 0.34 0.35 0.35 0.36 0.36 0.37 0.37 0.37 0.38 0.38 0.39 0.39 0.4  0.4
 0.41 0.41 0.42 0.42 0.43 0.43 0.44 0.44 0.45 0.45 0.46 0.46 0.46 0.47
 0.47 0.48]
```

In [38]:
```
#calculate distance
delta_t=dist(v_t,v_tp1)
delta_tm1=dist(v_tm1,v_t)
print("delta_T = ",delta_t,"delta_{T-1}=",delta_tm1)
```

```
delta_t =  6563611570.214574 delta_{t-1}= 5316525743.271799
```

Exercise 5.13.

In [39]:
```
c_mat = np.tile(W_vec.reshape(100,1),(1,100))-np.tile(W_vec.reshape(1,100)
c_pos = c_mat >0
c_mat[~c_pos] = 1e-7
u_mat = utility(c_mat)


v_prime = np.tile(v_tm1.reshape(1,100),(100,1))
v_prime[~c_pos]= -9e+4 #punish the upper diagonal entries ,not to choose t
v_tm2 = (u_mat + beta*v_prime).max(axis = 1)

print("V_{T-2}(W) is",v_tm2)

#confirm our analytical solution of psi(W)
index=np.argmax(u_mat+beta*v_prime,axis=1)
W_prime = W[index]
print("W'= psi-{T-2}(W)=",W_prime)
```

```
V_{T-2}(W) is [-8.10161181e+04 -7.29191115e+04 -6.56318055e+04 -1.24800112e+01
 -1.17868640e+01 -1.11630316e+01 -1.06015823e+01 -1.01961172e+01
```

```
 -9.83119864e+00 -9.50277190e+00 -9.21508983e+00 -8.95617596e+00
 -8.72315349e+00 -8.50000993e+00 -8.29918074e+00 -8.11685918e+00
 -7.93611290e+00 -7.77202350e+00 -7.61787282e+00 -7.47019236e+00
 -7.33145675e+00 -7.19792536e+00 -7.07306331e+00 -6.95288505e+00
 -6.83510202e+00 -6.72694159e+00 -6.62093686e+00 -6.51557634e+00
 -6.42017208e+00 -6.32486190e+00 -6.23003744e+00 -6.14302606e+00
 -6.05724690e+00 -5.97190488e+00 -5.89186218e+00 -5.81355194e+00
 -5.73635069e+00 -5.66224272e+00 -5.59020428e+00 -5.51972507e+00
 -5.45073219e+00 -5.38403502e+00 -5.31920043e+00 -5.25466191e+00
 -5.19256832e+00 -5.13194370e+00 -5.07191624e+00 -5.01383157e+00
 -4.95667316e+00 -4.90078893e+00 -4.84622677e+00 -4.79215955e+00
 -4.73988335e+00 -4.68844078e+00 -4.63714748e+00 -4.58804154e+00
 -4.53925138e+00 -4.49059088e+00 -4.44407086e+00 -4.39777255e+00
 -4.35160858e+00 -4.30715682e+00 -4.26324567e+00 -4.21945122e+00
 -4.17689161e+00 -4.13502359e+00 -4.09347602e+00 -4.05265403e+00
 -4.01264744e+00 -3.97312741e+00 -3.93390670e+00 -3.89560304e+00
 -3.85786272e+00 -3.82018150e+00 -3.78344171e+00 -3.74707406e+00
 -3.71106814e+00 -3.67576949e+00 -3.64067817e+00 -3.60620489e+00
 -3.57223859e+00 -3.53833704e+00 -3.50527122e+00 -3.47248140e+00
 -3.43975052e+00 -3.40798174e+00 -3.37623305e+00 -3.34465086e+00
 -3.31387920e+00 -3.28330953e+00 -3.25279814e+00 -3.22294517e+00
 -3.19343433e+00 -3.16397654e+00 -3.13498901e+00 -3.10641518e+00
 -3.07799121e+00 -3.04982033e+00 -3.02212584e+00 -2.99466558e+00]
W'= psi-{T-2}(W)= [0.01 0.01 0.02 0.03 0.03 0.04 0.05 0.05 0.06 0.07 0.07 0.08 0.09
 0.1  0.1  0.11 0.12 0.12 0.13 0.14 0.14 0.15 0.16 0.16 0.17 0.18 0.18
 0.19 0.19 0.2  0.2  0.21 0.22 0.22 0.23 0.24 0.24 0.25 0.26 0.26 0.27
 0.28 0.28 0.29 0.29 0.3  0.31 0.31 0.32 0.33 0.33 0.34 0.35 0.35 0.36
 0.36 0.37 0.37 0.38 0.39 0.39 0.4  0.41 0.41 0.42 0.43 0.43 0.44 0.45
 0.45 0.46 0.46 0.47 0.48 0.48 0.49 0.5  0.5  0.51 0.52 0.52 0.53 0.53
 0.54 0.55 0.55 0.56 0.56 0.57 0.58 0.58 0.59 0.6  0.6  0.61 0.62 0.62
 0.63 0.64]
```

```
In [40]: #calculate distance
         delta_t=dist(v_t,v_tp1)
         delta_tm1=dist(v_tm1,v_t)
         delta_tm2=dist(v_tm2,v_tm1)
         print("delta_T = ",delta_t,"delta_{T-1}=",delta_tm1,"delta_{T-2}=",delta_t
```

```
delta_T =  6563611570.214574 delta_{T-1}= 5316525743.271799 delta_{T-2}= 430638603
```

Exercise 5.14.

```
In [48]: maxiters = 500
         toler = 1e-9
         VF_iter = 0
         N=100
         beta=0.9
```

```
distance=10.0
#create utility matrix
c_mat = np.tile(W_vec.reshape(N,1),(1,N)) - np.tile(W_vec.reshape(1,N),(N,
c_pos = c_mat > 0
c_mat[~c_pos] = 1e-7 # replace the negative consumption
u_mat = utility(c_mat)
v_init = v_tp1
print("converging......")
while distance > toler and VF_iter < maxiters:
    VF_iter +=1
    v_prime = np.tile(v_init.reshape(1,N),(N,1))
    v_prime[~c_pos]= -9e+4 #punish the upper diagonal entries ,not to choo
    v_new = (u_mat + beta*v_prime).max(axis = 1)
    distance = dist(v_new,v_init)
    index=np.argmax(u_mat+beta*v_prime,axis=1)
    W_prime = W[index]
    print (VF_iter, distance)
    v_init = v_new
print("convergence completed")

print("V(W) = ",v_init,"\n", "it takes",VF_iter,"iterations" )
```

```
converging...
1 6563611570.214574
2 5316525743.271799
3 4306386030.006323
4 3488172794.571423
5 2825420037.630621
6 2288590282.5590854
7 1853758166.5375948
8 1501544142.7426844
9 1216250776.6422586
10 985163145.1419042
11 797982160.0373639
12 646365559.4266958
13 523556110.9935632
14 424080456.206154
15 343505174.74340326
16 278239195.885257
17 225373752.3122196
18 182552742.55996227
19 147867724.27005067
20 119772859.10622111
21 97016018.03859532
22 78582976.63315381
23 63652212.98545917
24 51558294.32776983
25 41762220.12006531
```

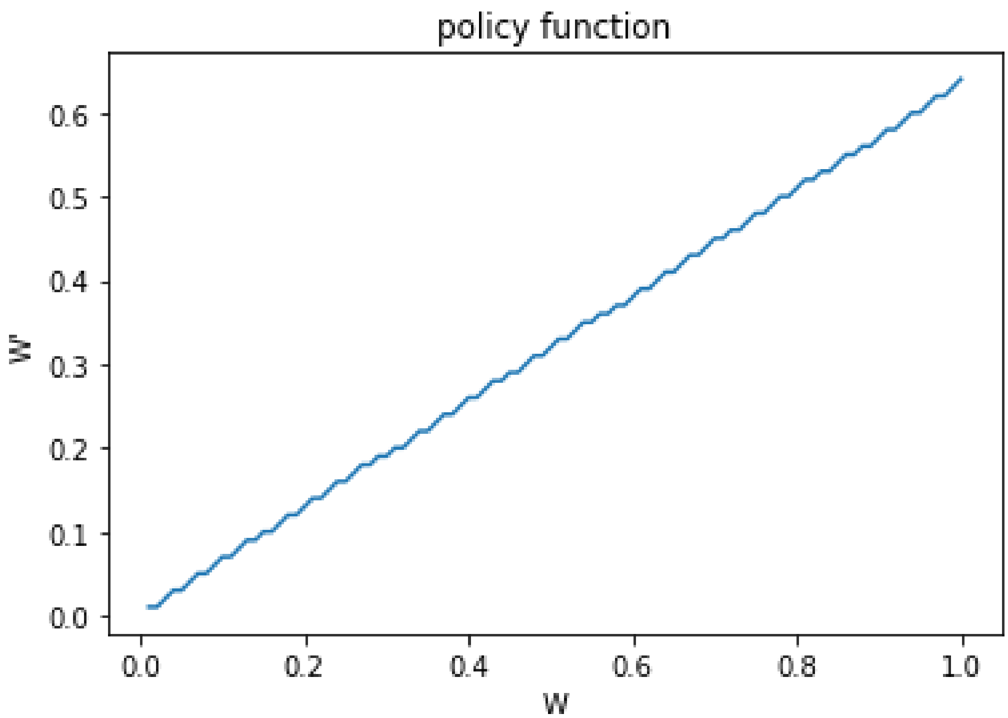
```
26 33827399.92370578
27 27400195.48398701
28 22194159.813998662
29 17977270.853317253
30 14561590.732883925
31 11794889.778709196
32 9553861.95285677
33 7738629.366687092
34 6268290.928397751
35 5077316.754233369
36 4112627.6374292998
37 3331229.420618223
38 2698296.8357309587
39 2185621.415306088
40 1770354.3006760497
41 1433987.9149444585
42 1161531.1223381404
43 940841.1015289262
44 762082.1677779292
45 617287.4156153648
46 500003.65221405274
47 405003.7907942261
48 328053.89114983805
49 265724.4605370663
50 215237.61132824747
51 174343.2533750371
52 141218.81426884214
53 114388.00935499245
54 92655.04917523319
55 75051.34382720976
56 60792.33427200551
57 49242.52936902186
58 39887.180071882685
59 32309.3403841846
60 26171.282961856
61 21199.449962701066
62 17172.257654033263
63 13910.225445246651
64 11267.97247896265
65 9127.741149256375
66 7394.146525049691
67 5989.928390551505
68 4852.503735570292
69 3931.1818695174766
70 3184.9040978319927
71 2580.410948027705
72 2090.7641764207274
73 1694.1410904514296
```

```
74 1372.867201451715
75 1112.627119310548
76 901.8230243640993
77 731.0600844858619
78 592.7310334744678
79 480.67457632861755
80 389.8971744723697
81 316.35332596891163
82 256.76931150496404
83 208.49415024915757
84 169.37141215479707
85 137.66542003422197
86 111.9686916994115
87 91.12990180160644
88 74.23008056632062
89 60.52291228596272
90 49.38428817377354
91 40.24315102906037
92 32.126682064026454
93 25.541799163125216
94 19.767115537050348
95 15.155025726039097
96 11.174263919022518
97 8.02000552946439
98 5.341082016591021
99 3.2347254868217576
100 1.4634529907462
101 0.0
convergence completed
V(W) =  [-8.10161181e+04 -7.29191115e+04 -6.56318055e+04 -5.90732301e+04
 -5.31705123e+04 -4.78580662e+04 -4.30768648e+04 -3.87737835e+04
 -3.49010103e+04 -3.14155144e+04 -2.82785681e+04 -2.54553165e+04
 -2.29143900e+04 -2.06275562e+04 -1.85694057e+04 -1.67170703e+04
 -1.50499685e+04 -1.35495768e+04 -1.21992243e+04 -1.09839070e+04
 -9.89012150e+03 -8.90571452e+03 -8.01974824e+03 -7.22237858e+03
 -6.50474589e+03 -5.85887647e+03 -5.27759400e+03 -4.75443977e+03
 -4.28360096e+03 -3.85984604e+03 -3.47846660e+03 -3.13522511e+03
 -2.82630777e+03 -2.54828216e+03 -2.29805912e+03 -2.07285838e+03
 -1.87017771e+03 -1.68776511e+03 -1.52359377e+03 -1.37583956e+03
 -1.24286077e+03 -1.12317987e+03 -1.01546705e+03 -9.18525516e+02
 -8.31278135e+02 -7.52755491e+02 -6.82085112e+02 -6.18481771e+02
 -5.61238764e+02 -5.09720058e+02 -4.63353223e+02 -4.21623071e+02
 -3.84065934e+02 -3.50264510e+02 -3.19843230e+02 -2.92464077e+02
 -2.67822839e+02 -2.45645726e+02 -2.25686323e+02 -2.07722861e+02
 -1.91555745e+02 -1.77005341e+02 -1.63909977e+02 -1.52124149e+02
 -1.41516905e+02 -1.31970384e+02 -1.23378516e+02 -1.15645835e+02
 -1.08686421e+02 -1.02422949e+02 -9.67858247e+01 -9.17124124e+01
 -8.71463414e+01 -8.30368774e+01 -7.93383599e+01 -7.60096941e+01
```

```
-7.30138948e+01 -7.03176755e+01 -6.78910782e+01 -6.57071405e+01
-6.37415967e+01 -6.19726072e+01 -6.03805167e+01 -5.89476352e+01
-5.76580418e+01 -5.64974078e+01 -5.54528372e+01 -5.45127237e+01
-5.36666215e+01 -5.29051296e+01 -5.22119824e+01 -5.15266396e+01
-5.09028071e+01 -5.02859987e+01 -4.97245494e+01 -4.91694218e+01
-4.86641175e+01 -4.81645026e+01 -4.77097288e+01 -4.72600754e+01]
it takes 101 iterations
```

Exercise 5.15.

```
In [56]: import matplotlib.pyplot as plt
         plt.plot(W_vec,W_prime)
         plt.xlabel("W")
         plt.ylabel("W'")
         plt.title("policy function")
         plt.show()
```



Exercise 5.16.

```
In [61]: #mu=1,sigma=0.5
         mu=1
         sig=0.5
         M=7
         epi=np.linspace(mu-3*sig,mu+3*sig,M)
```

```
print("support of epsilon is",epi)

from scipy.stats import norm

pdf = lambda x: norm(loc = mu, scale = sig).pdf(x)
pdf_e = pdf(epi)
print("pdf of epsilon is",pdf_e)
```

```
support of epsilon is [-0.5  0.   0.5  1.   1.5  2.   2.5]
pdf of epsilon is [0.0088637  0.10798193 0.48394145 0.79788456 0.48394145 0.1079819
 0.0088637 ]
```

Exercise 5.17.

```
In [96]: #set parameters and utility function
         N=100
         M=7
         beta=0.9
         def utility(c):
             ut = np.log(c)
             return ut

         #initial values
         #v_init=np.zeros((N,N,M))
         #W_vec = np.linspace(0.01,1,N)
         #c_mat = np.tile(W_vec.reshape(N,1),(N,1,M))-np.tile(W_vec.reshape(1,N),(1
         #c_pos = c_mat >0
         #c_mat[~c_pos] = 1e-7
         #u_mat = utility(c_mat)
         #e_mat = np.tile(epi.reshape(M,1),(N,N,1))
         #v_new = (u_mat*e_mat + beta*v_init).max(axis = 1)
```

```
In [132]: c_mat=np.tile(W_vec.reshape((N,1)), (1,N)) - np.tile(W_vec.reshape((1,N))
          c_pos=c_mat>0
          c_mat[~c_pos]=1e-7
          u_mat=utility(c_mat)
          eu_cube = np.array([u_mat*e for e in epi])
          V_init=np.zeros((N,M))
          EV = V_init @ pdf_e.reshape((M,1))
          EV_mat = np.tile(EV.reshape((1,N)), (N,1))
          EV_mat[~c_pos] = -9e+4
          EV_cube = np.array([EV_mat for e in range(M)])

          VT = eu_cube + beta*EV_cube
          V_new = np.zeros((N,M))
          W_new = np.zeros((N,M))
          for i in range(N):
              VTW = VT[:, i, :]
```

```
            V_new[i] = VTW.max(axis=1)
            ind = np.argmax(VTW, axis=1)
            W_new[i] = W_vec[ind]

        print(W_new)
```

```
[[0.01 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.01 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.02 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.03 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.04 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.05 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.06 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.07 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.08 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.09 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.1  0.01 0.01 0.01 0.01 0.01 0.01]
 [0.11 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.12 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.13 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.14 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.15 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.16 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.17 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.18 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.19 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.2  0.01 0.01 0.01 0.01 0.01 0.01]
 [0.21 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.22 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.23 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.24 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.25 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.26 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.27 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.28 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.29 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.3  0.01 0.01 0.01 0.01 0.01 0.01]
 [0.31 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.32 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.33 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.34 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.35 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.36 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.37 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.38 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.39 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.4  0.01 0.01 0.01 0.01 0.01 0.01]
 [0.41 0.01 0.01 0.01 0.01 0.01 0.01]
```

```
[0.42 0.01 0.01 0.01 0.01 0.01 0.01]
[0.43 0.01 0.01 0.01 0.01 0.01 0.01]
[0.44 0.01 0.01 0.01 0.01 0.01 0.01]
[0.45 0.01 0.01 0.01 0.01 0.01 0.01]
[0.46 0.01 0.01 0.01 0.01 0.01 0.01]
[0.47 0.01 0.01 0.01 0.01 0.01 0.01]
[0.48 0.01 0.01 0.01 0.01 0.01 0.01]
[0.49 0.01 0.01 0.01 0.01 0.01 0.01]
[0.5  0.01 0.01 0.01 0.01 0.01 0.01]
[0.51 0.01 0.01 0.01 0.01 0.01 0.01]
[0.52 0.01 0.01 0.01 0.01 0.01 0.01]
[0.53 0.01 0.01 0.01 0.01 0.01 0.01]
[0.54 0.01 0.01 0.01 0.01 0.01 0.01]
[0.55 0.01 0.01 0.01 0.01 0.01 0.01]
[0.56 0.01 0.01 0.01 0.01 0.01 0.01]
[0.57 0.01 0.01 0.01 0.01 0.01 0.01]
[0.58 0.01 0.01 0.01 0.01 0.01 0.01]
[0.59 0.01 0.01 0.01 0.01 0.01 0.01]
[0.6  0.01 0.01 0.01 0.01 0.01 0.01]
[0.61 0.01 0.01 0.01 0.01 0.01 0.01]
[0.62 0.01 0.01 0.01 0.01 0.01 0.01]
[0.63 0.01 0.01 0.01 0.01 0.01 0.01]
[0.64 0.01 0.01 0.01 0.01 0.01 0.01]
[0.65 0.01 0.01 0.01 0.01 0.01 0.01]
[0.66 0.01 0.01 0.01 0.01 0.01 0.01]
[0.67 0.01 0.01 0.01 0.01 0.01 0.01]
[0.68 0.01 0.01 0.01 0.01 0.01 0.01]
[0.69 0.01 0.01 0.01 0.01 0.01 0.01]
[0.7  0.01 0.01 0.01 0.01 0.01 0.01]
[0.71 0.01 0.01 0.01 0.01 0.01 0.01]
[0.72 0.01 0.01 0.01 0.01 0.01 0.01]
[0.73 0.01 0.01 0.01 0.01 0.01 0.01]
[0.74 0.01 0.01 0.01 0.01 0.01 0.01]
[0.75 0.01 0.01 0.01 0.01 0.01 0.01]
[0.76 0.01 0.01 0.01 0.01 0.01 0.01]
[0.77 0.01 0.01 0.01 0.01 0.01 0.01]
[0.78 0.01 0.01 0.01 0.01 0.01 0.01]
[0.79 0.01 0.01 0.01 0.01 0.01 0.01]
[0.8  0.01 0.01 0.01 0.01 0.01 0.01]
[0.81 0.01 0.01 0.01 0.01 0.01 0.01]
[0.82 0.01 0.01 0.01 0.01 0.01 0.01]
[0.83 0.01 0.01 0.01 0.01 0.01 0.01]
[0.84 0.01 0.01 0.01 0.01 0.01 0.01]
[0.85 0.01 0.01 0.01 0.01 0.01 0.01]
[0.86 0.01 0.01 0.01 0.01 0.01 0.01]
[0.87 0.01 0.01 0.01 0.01 0.01 0.01]
[0.88 0.01 0.01 0.01 0.01 0.01 0.01]
[0.89 0.01 0.01 0.01 0.01 0.01 0.01]
```

```
 [0.9  0.01 0.01 0.01 0.01 0.01 0.01]
 [0.91 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.92 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.93 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.94 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.95 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.96 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.97 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.98 0.01 0.01 0.01 0.01 0.01 0.01]
 [0.99 0.01 0.01 0.01 0.01 0.01 0.01]]
```

Exercise 5.18.

```
In [133]: def dist(v1,v2):
              return np.sum((v1-v2)**2)
          dis_t=dist(V_init,V_new)
          dis_t

Out[133]: 45945284542.69222
```

Exercise 5.19.

```
In [137]: V_init=V_new
          EV = V_init @ pdf_e.reshape((M,1))
          EV_mat = np.tile(EV.reshape((1,N)), (N,1))
          EV_mat[~c_pos] = -9e+4
          EV_cube = np.array([EV_mat for e in range(M)])

          VT = eu_cube + beta*EV_cube
          V_new_1 = np.zeros((N,M))
          W_new_1 = np.zeros((N,M))

          for i in range(N):
              VTW = VT[:, i, :]
              V_new_1[i] = VTW.max(axis=1)
              ind = np.argmax(VTW, axis=1)

              W_new_1[i] = W_vec[ind]

In [138]: dis_tm1=dist(V_new_1,V_new)
          print("delta T:",dis_t)
          print("Delta T-1:",dis_tm1)

delta T: 45945284542.69222
Delta T-1: 45940067294.25311
```

Exercise 5.20.

```
In [139]: V_init=V_new_1
          EV = V_init @ pdf_e.reshape((M,1))
          EV_mat = np.tile(EV.reshape((1,N)), (N,1))
          EV_mat[~c_pos] = -9e+4
          EV_cube = np.array([EV_mat for e in range(M)])

          VT = eu_cube + beta*EV_cube
          V_new_2 = np.zeros((N,M))
          W_new_2 = np.zeros((N,M))
          for i in range(N):
              VTW = VT[:, i, :]
              V_new_2[i] = VTW.max(axis=1)
              ind = np.argmax(VTW, axis=1)
              #ind_list.append(ind)
              W_new_2[i] = W_vec[ind]

In [140]: dis_tm2=dist(V_new_2,V_new_1)
          print("delta T:",dis_t)
          print("delta T-1:",dis_tm1)
          print("delta T-2:",dis_tm2)

delta T: 45945284542.69222
delta T-1: 45940067294.25311
delta T-2: 45930697106.37403
```

Exercise 5.21.

```
In [148]: VF_iter = 0
          distance=10
          V_init = np.zeros((N,M))
          c_mat=np.tile(W_vec.reshape((N,1)), (1,N)) - np.tile(W_vec.reshape((1,N))
          c_pos=c_mat>0
          c_mat[~c_pos]=1e-7
          u_mat=utility(c_mat)
          eu_cube = np.array([u_mat*e for e in epi])

          while distance>toler and VF_iter<maxiters:
              VF_iter += 1
              EV = V_init @ pdf_e.reshape((M,1))
             # v_exp_mat = np.tile(v_exp.reshape((1, N)), (N, 1))
              EV_mat = np.tile(EV.reshape(1,N), (N,1))
              EV_mat[~c_pos] = -9e+4
              EV_cube = np.array([EV_mat for e in range(M)])

              V = eu_cube + beta*EV_cube
              V_new = np.zeros((N,M))
              W_new = np.zeros((N,M))
              for i in range(N):
```

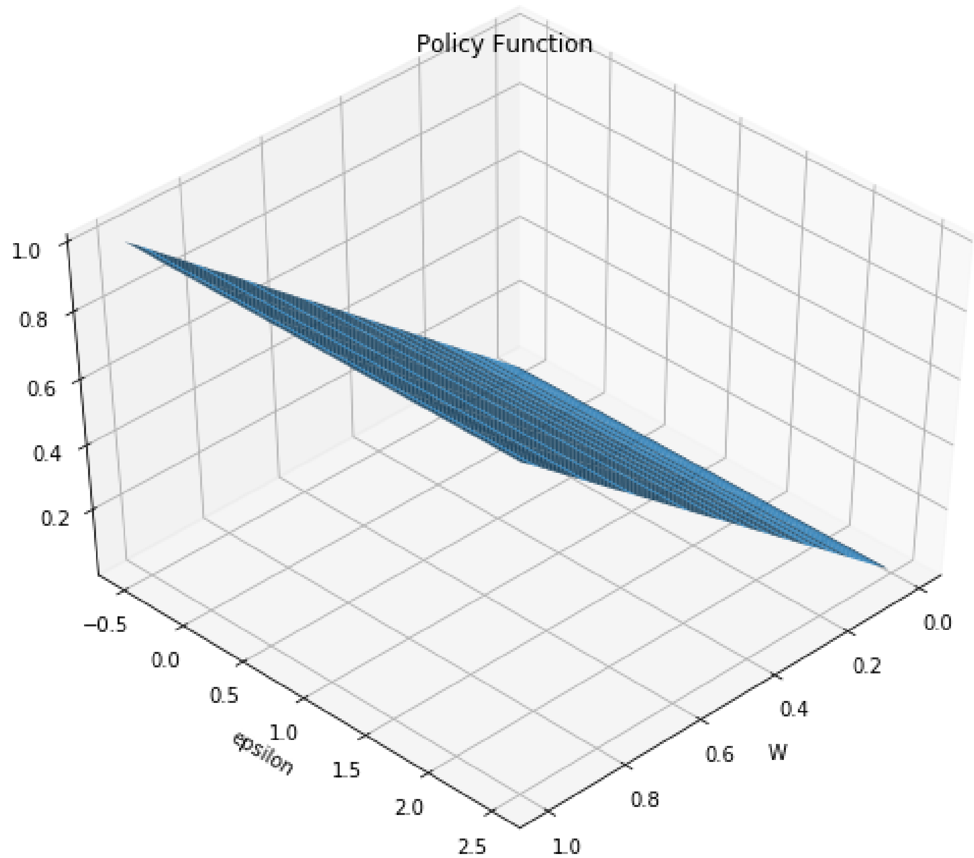
```
                VTW = V[:, i, :]
                V_new[i] = VTW.max(axis=1)
                ind = np.argmax(VTW, axis=1)
                W_new[i] = W_vec[ind]
            distance = np.sum((V_init-V_new)**2)
            print(VF_iter, distance)
            V_init = V_new
        print("convergence completed")
        print("it takes",VF_iter,"iterations" )
```

```
1 45945284542.69222
2 45940067294.25311
3 45930697106.37403
4 45913879281.491066
5 45883756170.673996
6 45829999996.0005
7 45734713997.20003
8 45567917314.08906
9 45282852628.96708
10 44818584349.20162
11 44140197767.835655
12 43426000016.028946
13 43774426087.622765
14 49669342029.82887
15 77263245984.69931
16 181683997734.36615
17 416333292134.422
18 315005356423.8015
19 332719741.4032231
20 0.0
convergence completed
it takes 20 iterations
```

In [ ]:

Exercise 5.22.

In [149]:
```
from mpl_toolkits.mplot3d import Axes3D
X, Y = np.meshgrid(W_vec, epi)
new_fig = plt.figure(figsize=(10,8))
new_ax = new_fig.add_subplot(111, projection='3d')
new_ax.plot_surface(X.T, Y.T, W_new)
new_ax.set_xlabel('W')
new_ax.set_ylabel('epsilon')
new_ax.set_title("Policy Function")
new_ax.view_init(elev=45,azim=45)
plt.show()
```



Policy Function
1.0
0.8
0.6
0.4
0.2
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
epsilon
0.0
0.2
0.4
0.6
0.8
1.0
W


```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```