ProblemSet2_ANSWER

May 15, 2019

0.0.1 Problem Set 2

MACS 30250, Dr. Evans

Due Wednesday, May. 15 at 1:30pm

Tianxin Zheng

1.Stochastic i.i.d. cake eating problem

In [1]:
```
import numpy as np
import scipy.optimize as opt
import scipy.interpolate as intpl
import matplotlib.pyplot as plt
%matplotlib notebook
```

(a)

In [2]:
```
# Set up the parameters
beta = 0.9
gamma = 2.2
W_min = 0.1
W_max = 10.0
W_size = 30
W_vec = np.linspace(W_min, W_max, W_size)
V_t = np.log(W_vec)

def util_CRRA(W, W_pr, gamma):
    # Define CRRA utility function
    c = W - W_pr
    util = (c ** (1 - gamma) - 1) / (1 - gamma)

    return util

eps_vec = np.array([-1.40, -0.55,0.00,0.55,1.4])
eps_prob = np.array([0.1,0.2,0.4,0.2,0.1])
eps_size = eps_vec.shape[0]
```

```
        def neg_V_iid(W_pr, *args):
            W_init, eps, util, EXP_V_t_interp, gamma, beta = args
            Vtp1 = np.exp(eps) * util(W, W_pr, gamma) + beta * EXP_V_t_interp(W_pr)
            neg_Vtp1 = -Vtp1

            return neg_Vtp1

In [3]: V_init = np.zeros((W_size, eps_size))
        V_new = V_init.copy()

        VF_iter = 0
        VF_dist = 10
        VF_maxiter = 200
        VF_mindist = 1e-8

        while (VF_iter < VF_maxiter) and (VF_dist > VF_mindist):
            VF_iter += 1
            V_init = V_new.copy()
            V_new = np.zeros((W_size, eps_size))
            psi_vec = np.zeros((W_size, eps_size))

            # Intergrate out eps_pr from V_init
            Exp_V = V_init @ eps_prob.reshape((eps_size, 1))

            # Interpolate ? value function
            Exp_V_interp = intpl.interp1d(W_vec, Exp_V.flatten(), kind='cubic',
                                          fill_value='extrapolate')

            for eps_ind in range(eps_size):
                for W_ind in range(W_size):
                    W = W_vec[W_ind]
                    eps = eps_vec[eps_ind]
                    V_args = (W, eps, util_CRRA, Exp_V_interp, gamma, beta)
                    results_all = opt.minimize_scalar(neg_V_iid, bounds=(1e-10, W - 1e-10),
                                                      args=V_args, method='bounded')
                    V_new[W_ind, eps_ind] = -results_all.fun
                    psi_vec[W_ind, eps_ind] = results_all.x

            VF_dist = ((V_init - V_new) ** 2).sum()
            print('VF_iter=', VF_iter, ', VF_dist=', VF_dist)

VF_iter= 1 , VF_dist= 3494.416552492849
VF_iter= 2 , VF_dist= 3288.9775602179398
VF_iter= 3 , VF_dist= 4368.033199294504
VF_iter= 4 , VF_dist= 5171.123172733101
VF_iter= 5 , VF_dist= 5690.816865389136
VF_iter= 6 , VF_dist= 5962.334129328155
VF_iter= 7 , VF_dist= 6029.410863441401
```

```
VF_iter= 8 , VF_dist= 5938.006359170556
VF_iter= 9 , VF_dist= 5728.733568262043
VF_iter= 10 , VF_dist= 5436.089591828773
VF_iter= 11 , VF_dist= 5088.450567375278
VF_iter= 12 , VF_dist= 4708.595711207952
VF_iter= 13 , VF_dist= 4314.425870978352
VF_iter= 14 , VF_dist= 3919.6529865086823
VF_iter= 15 , VF_dist= 3534.453890340005
VF_iter= 16 , VF_dist= 3166.0717799094914
VF_iter= 17 , VF_dist= 2819.3724059115366
VF_iter= 18 , VF_dist= 2497.331372214219
VF_iter= 19 , VF_dist= 2201.4557625475145
VF_iter= 20 , VF_dist= 1932.1399178953566
VF_iter= 21 , VF_dist= 1688.9596638065682
VF_iter= 22 , VF_dist= 1470.9108719157662
VF_iter= 23 , VF_dist= 1276.5993702275612
VF_iter= 24 , VF_dist= 1104.3893159739218
VF_iter= 25 , VF_dist= 952.5168794056659
VF_iter= 26 , VF_dist= 819.1754636558386
VF_iter= 27 , VF_dist= 702.5782302191728
VF_iter= 28 , VF_dist= 601.0007639998626
VF_iter= 29 , VF_dist= 512.8120804049411
VF_iter= 30 , VF_dist= 436.49195996903325
VF_iter= 31 , VF_dist= 370.64094039451504
VF_iter= 32 , VF_dist= 313.9837850772325
VF_iter= 33 , VF_dist= 265.36837905541597
VF_iter= 34 , VF_dist= 223.76142626173007
VF_iter= 35 , VF_dist= 188.24202140834456
VF_iter= 36 , VF_dist= 157.99391929400753
VF_iter= 37 , VF_dist= 132.29712064193444
VF_iter= 38 , VF_dist= 110.51922993676429
VF_iter= 39 , VF_dist= 92.10691083376445
VF_iter= 40 , VF_dist= 76.57766324529271
VF_iter= 41 , VF_dist= 63.512068135772914
VF_iter= 42 , VF_dist= 52.54658698138166
VF_iter= 43 , VF_dist= 43.3669590697024
VF_iter= 44 , VF_dist= 35.702208079202634
VF_iter= 45 , VF_dist= 29.31924705157791
VF_iter= 46 , VF_dist= 24.018055757038024
VF_iter= 47 , VF_dist= 19.62739479941466
VF_iter= 48 , VF_dist= 16.001015214233433
VF_iter= 49 , VF_dist= 13.014319735746323
VF_iter= 50 , VF_dist= 10.56143149616109
VF_iter= 51 , VF_dist= 8.552627061405703
VF_iter= 52 , VF_dist= 6.912092964935729
VF_iter= 53 , VF_dist= 5.575967859534671
VF_iter= 54 , VF_dist= 4.490635846895642
VF_iter= 55 , VF_dist= 3.611240167867626
```

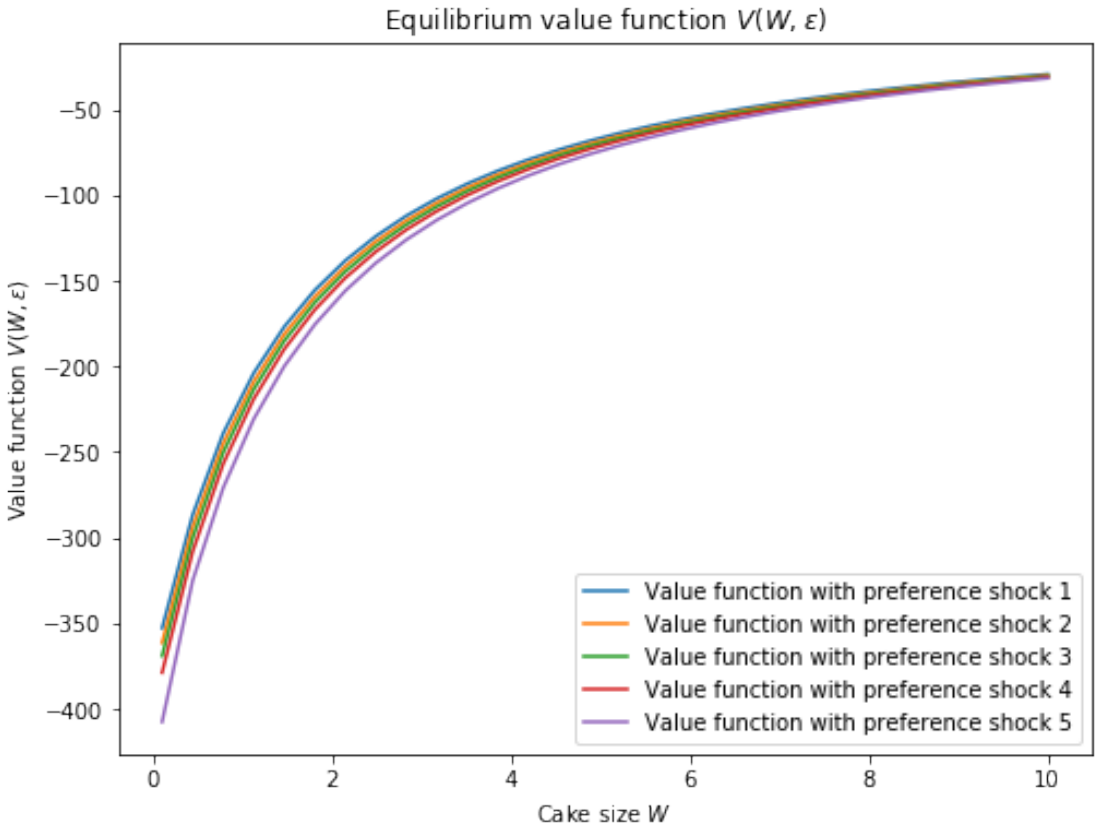
```
VF_iter= 56 , VF_dist= 2.9003900724273053
VF_iter= 57 , VF_dist= 2.3270371537171513
VF_iter= 58 , VF_dist= 1.8655005992880302
VF_iter= 59 , VF_dist= 1.4946235941207497
VF_iter= 60 , VF_dist= 1.1970454632792984
VF_iter= 61 , VF_dist= 0.9585760728643425
VF_iter= 62 , VF_dist= 0.7676605629606206
VF_iter= 63 , VF_dist= 0.6149237330662378
VF_iter= 64 , VF_dist= 0.4927844192420875
VF_iter= 65 , VF_dist= 0.3951310655441188
VF_iter= 66 , VF_dist= 0.31705046239942836
VF_iter= 67 , VF_dist= 0.25460234391745856
VF_iter= 68 , VF_dist= 0.20463323107554093
VF_iter= 69 , VF_dist= 0.16462358821635545
VF_iter= 70 , VF_dist= 0.1325630262736468
VF_iter= 71 , VF_dist= 0.10684893010182255
VF_iter= 72 , VF_dist= 0.08620449950960188
VF_iter= 73 , VF_dist= 0.06961276460117397
VF_iter= 74 , VF_dist= 0.05626365791980316
VF_iter= 75 , VF_dist= 0.045511694260017735
VF_iter= 76 , VF_dist= 0.03684222157750761
VF_iter= 77 , VF_dist= 0.029844564402374686
VF_iter= 78 , VF_dist= 0.024190686856972787
VF_iter= 79 , VF_dist= 0.019618260386259922
VF_iter= 80 , VF_dist= 0.015917236304987616
VF_iter= 81 , VF_dist= 0.012919200660435864
VF_iter= 82 , VF_dist= 0.010488934004746101
VF_iter= 83 , VF_dist= 0.008517716355044384
VF_iter= 84 , VF_dist= 0.00691801246777643
VF_iter= 85 , VF_dist= 0.005619248530286449
VF_iter= 86 , VF_dist= 0.004564451958587326
VF_iter= 87 , VF_dist= 0.0037075740906608633
VF_iter= 88 , VF_dist= 0.0030113536420462708
VF_iter= 89 , VF_dist= 0.0024456088523578355
VF_iter= 90 , VF_dist= 0.001985869936846679
VF_iter= 91 , VF_dist= 0.0016122821021656536
VF_iter= 92 , VF_dist= 0.0013087240486581234
VF_iter= 93 , VF_dist= 0.0010620984074924253
VF_iter= 94 , VF_dist= 0.0008617596247306495
VF_iter= 95 , VF_dist= 0.0006990519398148664
VF_iter= 96 , VF_dist= 0.0005669357206090445
VF_iter= 97 , VF_dist= 0.0004596848484600202
VF_iter= 98 , VF_dist= 0.0003726413433400984
VF_iter= 99 , VF_dist= 0.00030201618674586747
VF_iter= 100 , VF_dist= 0.00024472749354110894
VF_iter= 101 , VF_dist= 0.00019826892637177278
VF_iter= 102 , VF_dist= 0.00016060263224256825
VF_iter= 103 , VF_dist= 0.0001300720891483579
```

```
VF_iter= 104 , VF_dist= 0.00010533113515101626
VF_iter= 105 , VF_dist= 8.528616230351807e-05
VF_iter= 106 , VF_dist= 6.904902893045537e-05
VF_iter= 107 , VF_dist= 5.58987032119717e-05
VF_iter= 108 , VF_dist= 4.525002255832318e-05
VF_iter= 109 , VF_dist= 3.662825367946391e-05
VF_iter= 110 , VF_dist= 2.9648381684870055e-05
VF_iter= 111 , VF_dist= 2.39982549749399e-05
VF_iter= 112 , VF_dist= 1.9424873004279235e-05
VF_iter= 113 , VF_dist= 1.5723235673077267e-05
VF_iter= 114 , VF_dist= 1.272727964557564e-05
VF_iter= 115 , VF_dist= 1.0302514478394718e-05
VF_iter= 116 , VF_dist= 8.34004242997388e-06
VF_iter= 117 , VF_dist= 6.751704204548888e-06
VF_iter= 118 , VF_dist= 5.466140377418238e-06
VF_iter= 119 , VF_dist= 4.425597146845334e-06
VF_iter= 120 , VF_dist= 3.5833367766181355e-06
VF_iter= 121 , VF_dist= 2.901539092470968e-06
VF_iter= 122 , VF_dist= 2.3496014077785377e-06
VF_iter= 123 , VF_dist= 1.9027617856574288e-06
VF_iter= 124 , VF_dist= 1.5409843211358737e-06
VF_iter= 125 , VF_dist= 1.2480569000018439e-06
VF_iter= 126 , VF_dist= 1.0108610126543219e-06
VF_iter= 127 , VF_dist= 8.187809298941962e-07
VF_iter= 128 , VF_dist= 6.632257181555538e-07
VF_iter= 129 , VF_dist= 5.372425577246577e-07
VF_iter= 130 , VF_dist= 4.352039711978974e-07
VF_iter= 131 , VF_dist= 3.52554804532906e-07
VF_iter= 132 , VF_dist= 2.856075891862148e-07
VF_iter= 133 , VF_dist= 2.3137700024210983e-07
VF_iter= 134 , VF_dist= 1.8744595280674418e-07
VF_iter= 135 , VF_dist= 1.518572697400391e-07
VF_iter= 136 , VF_dist= 1.2302604725753922e-07
VF_iter= 137 , VF_dist= 9.966874322757488e-08
VF_iter= 138 , VF_dist= 8.074580018350248e-08
VF_iter= 139 , VF_dist= 6.541520832632279e-08
VF_iter= 140 , VF_dist= 5.299492534475509e-08
VF_iter= 141 , VF_dist= 4.293245385716458e-08
VF_iter= 142 , VF_dist= 3.4780217313545753e-08
VF_iter= 143 , VF_dist= 2.817561810545218e-08
VF_iter= 144 , VF_dist= 2.282489479627886e-08
VF_iter= 145 , VF_dist= 1.8490046714220514e-08
VF_iter= 146 , VF_dist= 1.4978247147702045e-08
VF_iter= 147 , VF_dist= 1.2133266004667127e-08
VF_iter= 148 , VF_dist= 9.828523946527788e-09
```

(b)

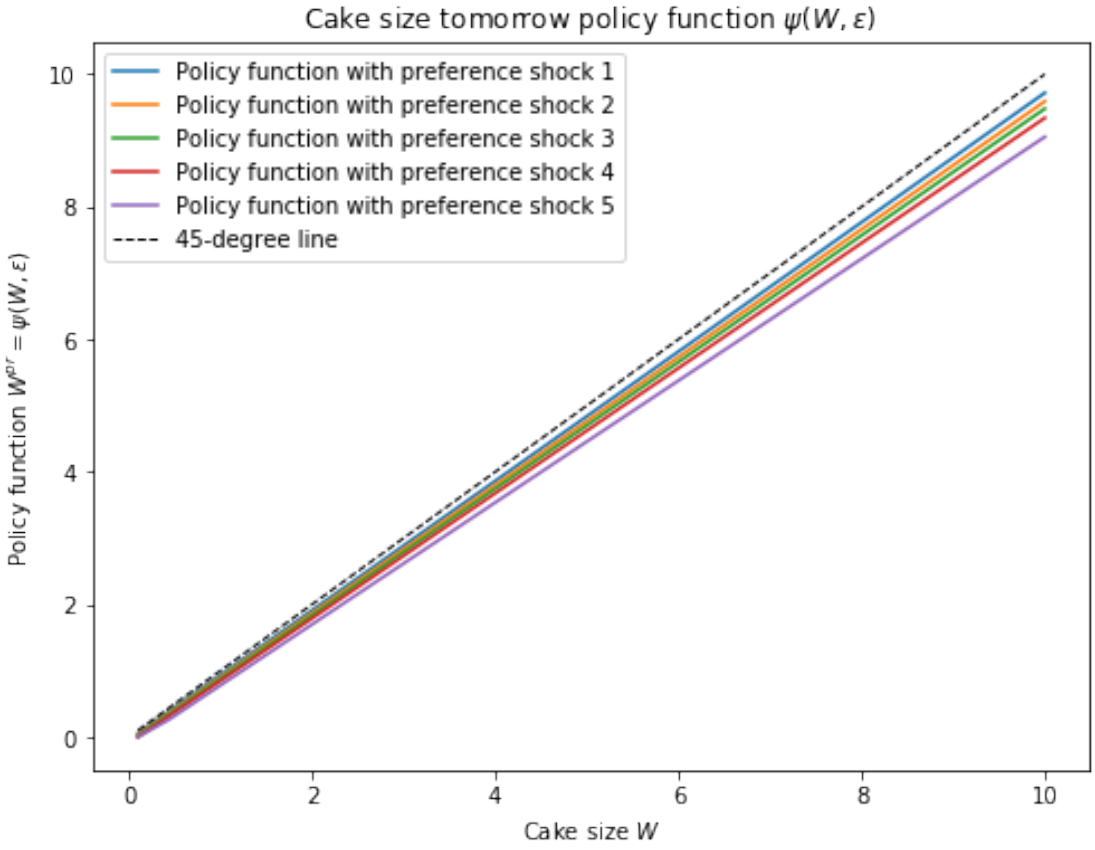
```
In [4]: plt.figure(figsize=(8,6))
        for i in range(5):
            plt.plot(W_vec, V_new[:,i], label='Value function with preference shock {}'.format
        plt.title('Equilibrium value function $V(W, \epsilon)$')
        plt.xlabel('Cake size $W$')
        plt.ylabel('Value function $V(W, \epsilon)$')
        plt.legend()
```

```
Out[4]: <matplotlib.legend.Legend at 0x1108c5198>
```




```
In [5]: plt.figure(figsize=(8,6))
        for i in range(5):
            plt.plot(W_vec, psi_vec[:,i], label='Policy function with preference shock {}'.for
        plt.plot(W_vec, W_vec, color = 'black', linewidth=1, linestyle='--',
                 label='45-degree line')
        plt.title('Cake size tomorrow policy function $\psi(W, \epsilon)$')
        plt.xlabel('Cake size $W$')
        plt.ylabel('Policy function $W^{pr}=\psi(W, \epsilon)$')
        plt.legend()
```

```
Out[5]: <matplotlib.legend.Legend at 0x1109dd748>
```



2. Persistent AR(1) stochastic cake eating problem

(a)

```
In [6]: trans_mat=np.array([[0.40, 0.28, 0.18, 0.10, 0.04],
                            [0.20, 0.40, 0.20, 0.13, 0.07],
                            [0.10, 0.20, 0.40, 0.20, 0.10],
                            [0.07, 0.13, 0.20, 0.40, 0.20],
                            [0.04, 0.10, 0.18, 0.28, 0.40]])

In [7]: V_init = np.zeros((W_size,eps_size))

        V_new = V_init.copy()

        VF_iter = 0
        VF_dist = 10
        VF_maxiter = 200
        VF_mindist = 1e-8
```

```
        while (VF_iter < VF_maxiter) and (VF_dist > VF_mindist):
            VF_iter += 1
            V_init = V_new.copy()
            V_new = np.zeros((W_size,eps_size))
            psi_mat = np.zeros((W_size,eps_size))
            for eps_ind in range(eps_size):
                eps = eps_vec[eps_ind]
                eps_prob = trans_mat[eps_ind,:]
                Exp_V = V_init @ eps_prob.reshape((eps_size,1))
                Exp_V_interp = intpl.interp1d(W_vec, Exp_V.flatten(), kind='cubic',
                                              fill_value='extrapolate')
                for W_ind in range(W_size):
                    W = W_vec[W_ind]
                    V_args = (W, eps, util_CRRA, Exp_V_interp, gamma, beta)
                    results1 = opt.minimize_scalar(neg_V_iid, bounds=(1e-10, W - 1e-10),
                                                   args=V_args, method='bounded')
                    V_new[W_ind, eps_ind] = -results1.fun
                    psi_mat[W_ind, eps_ind] = results1.x

            VF_dist = ((V_init - V_new) ** 2).sum()
            print('VF_iter=', VF_iter, ', VF_dist=', VF_dist)
```

```
VF_iter= 1 , VF_dist= 3494.416552492849
VF_iter= 2 , VF_dist= 4874.985394413625
VF_iter= 3 , VF_dist= 5656.12392067181
VF_iter= 4 , VF_dist= 6339.326577448594
VF_iter= 5 , VF_dist= 6784.60512520715
VF_iter= 6 , VF_dist= 6985.799159895715
VF_iter= 7 , VF_dist= 6981.428377619099
VF_iter= 8 , VF_dist= 6816.840486149343
VF_iter= 9 , VF_dist= 6534.059682039727
VF_iter= 10 , VF_dist= 6169.1406209651495
VF_iter= 11 , VF_dist= 5751.815350888644
VF_iter= 12 , VF_dist= 5305.8877157047045
VF_iter= 13 , VF_dist= 4849.904814536473
VF_iter= 14 , VF_dist= 4397.907932627158
VF_iter= 15 , VF_dist= 3960.1652494132745
VF_iter= 16 , VF_dist= 3543.8552910466856
VF_iter= 17 , VF_dist= 3153.675483893924
VF_iter= 18 , VF_dist= 2792.3708357696564
VF_iter= 19 , VF_dist= 2461.1834460824566
VF_iter= 20 , VF_dist= 2160.226185380022
VF_iter= 21 , VF_dist= 1888.7889925601917
VF_iter= 22 , VF_dist= 1645.5852440446392
VF_iter= 23 , VF_dist= 1428.9469626205291
VF_iter= 24 , VF_dist= 1236.97650716786
VF_iter= 25 , VF_dist= 1067.6628129910018
```

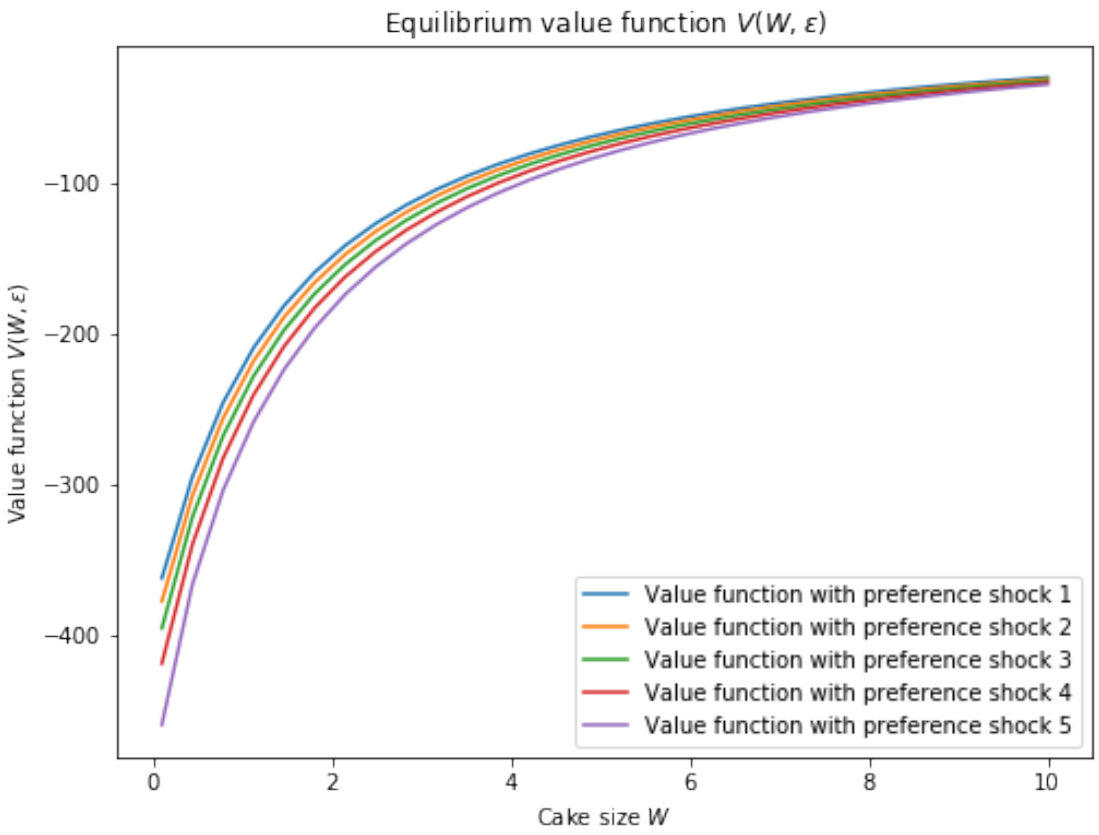
```
VF_iter= 26 , VF_dist= 918.9679255472375
VF_iter= 27 , VF_dist= 788.8896288130811
VF_iter= 28 , VF_dist= 675.5057004824748
VF_iter= 29 , VF_dist= 577.0027218827461
VF_iter= 30 , VF_dist= 491.6940988700029
VF_iter= 31 , VF_dist= 418.02889741622175
VF_iter= 32 , VF_dist= 354.59458555108137
VF_iter= 33 , VF_dist= 300.1149965095527
VF_iter= 34 , VF_dist= 253.44524684720494
VF_iter= 35 , VF_dist= 213.56437808664515
VF_iter= 36 , VF_dist= 179.56676835650512
VF_iter= 37 , VF_dist= 150.65287514806832
VF_iter= 38 , VF_dist= 126.11977523378562
VF_iter= 39 , VF_dist= 105.3518322902685
VF_iter= 40 , VF_dist= 87.81172046179698
VF_iter= 41 , VF_dist= 73.03195323589003
VF_iter= 42 , VF_dist= 60.607007263379
VF_iter= 43 , VF_dist= 50.186086183324804
VF_iter= 44 , VF_dist= 41.46653693273445
VF_iter= 45 , VF_dist= 34.18790792785717
VF_iter= 46 , VF_dist= 28.126622690158456
VF_iter= 47 , VF_dist= 23.091232274657113
VF_iter= 48 , VF_dist= 18.918203908566724
VF_iter= 49 , VF_dist= 15.468200429360907
VF_iter= 50 , VF_dist= 12.622804626463294
VF_iter= 51 , VF_dist= 10.28164372636001
VF_iter= 52 , VF_dist= 8.359871522439507
VF_iter= 53 , VF_dist= 6.785968609514036
VF_iter= 54 , VF_dist= 5.499824513152387
VF_iter= 55 , VF_dist= 4.451068970557403
VF_iter= 56 , VF_dist= 3.5976230220210104
VF_iter= 57 , VF_dist= 2.9044437903776936
VF_iter= 58 , VF_dist= 2.34243977518304
VF_iter= 59 , VF_dist= 1.8875361281043217
VF_iter= 60 , VF_dist= 1.519871703428028
VF_iter= 61 , VF_dist= 1.223111706456853
VF_iter= 62 , VF_dist= 0.9838615299677128
VF_iter= 63 , VF_dist= 0.791168913644621
VF_iter= 64 , VF_dist= 0.6361029250474338
VF_iter= 65 , VF_dist= 0.5113994806198763
VF_iter= 66 , VF_dist= 0.4111642302174867
VF_iter= 67 , VF_dist= 0.3306246409247078
VF_iter= 68 , VF_dist= 0.2659240496514192
VF_iter= 69 , VF_dist= 0.2139513170948042
VF_iter= 70 , VF_dist= 0.1722005119776325
VF_iter= 71 , VF_dist= 0.1386557855182209
VF_iter= 72 , VF_dist= 0.11169726169362215
VF_iter= 73 , VF_dist= 0.09002436947782042
```

```
VF_iter= 74 , VF_dist= 0.07259357958861436
VF_iter= 75 , VF_dist= 0.058567982235562074
VF_iter= 76 , VF_dist= 0.047276556890749444
VF_iter= 77 , VF_dist= 0.03818134393631556
VF_iter= 78 , VF_dist= 0.030851035730159
VF_iter= 79 , VF_dist= 0.02493976610538183
VF_iter= 80 , VF_dist= 0.02017009765618433
VF_iter= 81 , VF_dist= 0.01631939040584138
VF_iter= 82 , VF_dist= 0.013208888482201654
VF_iter= 83 , VF_dist= 0.010694987666865507
VF_iter= 84 , VF_dist= 0.008662250304210261
VF_iter= 85 , VF_dist= 0.0070178185919089545
VF_iter= 86 , VF_dist= 0.005686946025382914 5
VF_iter= 87 , VF_dist= 0.004609422370120485
VF_iter= 88 , VF_dist= 0.0037367124191735324
VF_iter= 89 , VF_dist= 0.0030296648670472557
VF_iter= 90 , VF_dist= 0.0024566765881688856
VF_iter= 91 , VF_dist= 0.001992220776133969
VF_iter= 92 , VF_dist= 0.00161566592165916 4
VF_iter= 93 , VF_dist= 0.0013103273925867242
VF_iter= 94 , VF_dist= 0.0010627051579994561
VF_iter= 95 , VF_dist= 0.0008618705959112186
VF_iter= 96 , VF_dist= 0.000698972799180187
VF_iter= 97 , VF_dist= 0.0005668407509089944
VF_iter= 98 , VF_dist= 0.0004596624861048065
VF_iter= 99 , VF_dist= 0.00037272613098643696
VF_iter= 100 , VF_dist= 0.0003022107274540966
VF_iter= 101 , VF_dist= 0.0002450171492676402
VF_iter= 102 , VF_dist= 0.0001986313339914293
VF_iter= 103 , VF_dist= 0.00016101358675039886
VF_iter= 104 , VF_dist= 0.00013050893615279547
VF_iter= 105 , VF_dist= 0.00010577450265560843
VF_iter= 106 , VF_dist= 8.572062460748406e-05
VF_iter= 107 , VF_dist= 6.946311911047963e-05
VF_iter= 108 , VF_dist= 5.6284558602380516e-05
VF_iter= 109 , VF_dist= 4.5602852794376086e-05
VF_iter= 110 , VF_dist= 3.6945752499598345e-05
VF_iter= 111 , VF_dist= 2.9930155676384144e-05
VF_iter= 112 , VF_dist= 2.4245309663471903e-05
VF_iter= 113 , VF_dist= 1.9639175069992804e-05
VF_iter= 114 , VF_dist= 1.5907355728518072e-05
VF_iter= 115 , VF_dist= 1.2884112283122067e-05
VF_iter= 116 , VF_dist= 1.0435067214135358e-05
VF_iter= 117 , VF_dist= 8.451283519075744e-06
VF_iter= 118 , VF_dist= 6.844458674015574e-06
VF_iter= 119 , VF_dist= 5.543024358491157e-06
VF_iter= 120 , VF_dist= 4.488981551249566e-06
VF_iter= 121 , VF_dist= 3.6353327310852107e-06
```

```
VF_iter= 122 , VF_dist= 2.943998810995179e-06
VF_iter= 123 , VF_dist= 2.3841295554757415e-06
VF_iter= 124 , VF_dist= 1.9307333682581294e-06
VF_iter= 125 , VF_dist= 1.5635662339836826e-06
VF_iter= 126 , VF_dist= 1.2662309477630227e-06
VF_iter= 127 , VF_dist= 1.0254469301659195e-06
VF_iter= 128 , VF_dist= 8.304583962909381e-07
VF_iter= 129 , VF_dist= 6.725547457491646e-07
VF_iter= 130 , VF_dist= 5.446819406192471e-07
VF_iter= 131 , VF_dist= 4.4112761496225405e-07
VF_iter= 132 , VF_dist= 3.5726599797105377e-07
VF_iter= 133 , VF_dist= 2.8935128587648305e-07
VF_iter= 134 , VF_dist= 2.3435026815796068e-07
VF_iter= 135 , VF_dist= 1.8980678277275483e-07
VF_iter= 136 , VF_dist= 1.5373192912320232e-07
VF_iter= 137 , VF_dist= 1.245151713786484e-07
VF_iter= 138 , VF_dist= 1.0085235263566314e-07
VF_iter= 139 , VF_dist= 8.168739194090008e-08
VF_iter= 140 , VF_dist= 6.616508779991977e-08
VF_iter= 141 , VF_dist= 5.359289518903287e-08
VF_iter= 142 , VF_dist= 4.340998180457873e-08
VF_iter= 143 , VF_dist= 3.51621655905898e-08
VF_iter= 144 , VF_dist= 2.8481629018835357e-08
VF_iter= 145 , VF_dist= 2.3070489617555775e-08
VF_iter= 146 , VF_dist= 1.868749732046429e-08
VF_iter= 147 , VF_dist= 1.5137265955297337e-08
VF_iter= 148 , VF_dist= 1.2261547883836958e-08
VF_iter= 149 , VF_dist= 9.93217479503226e-09
```

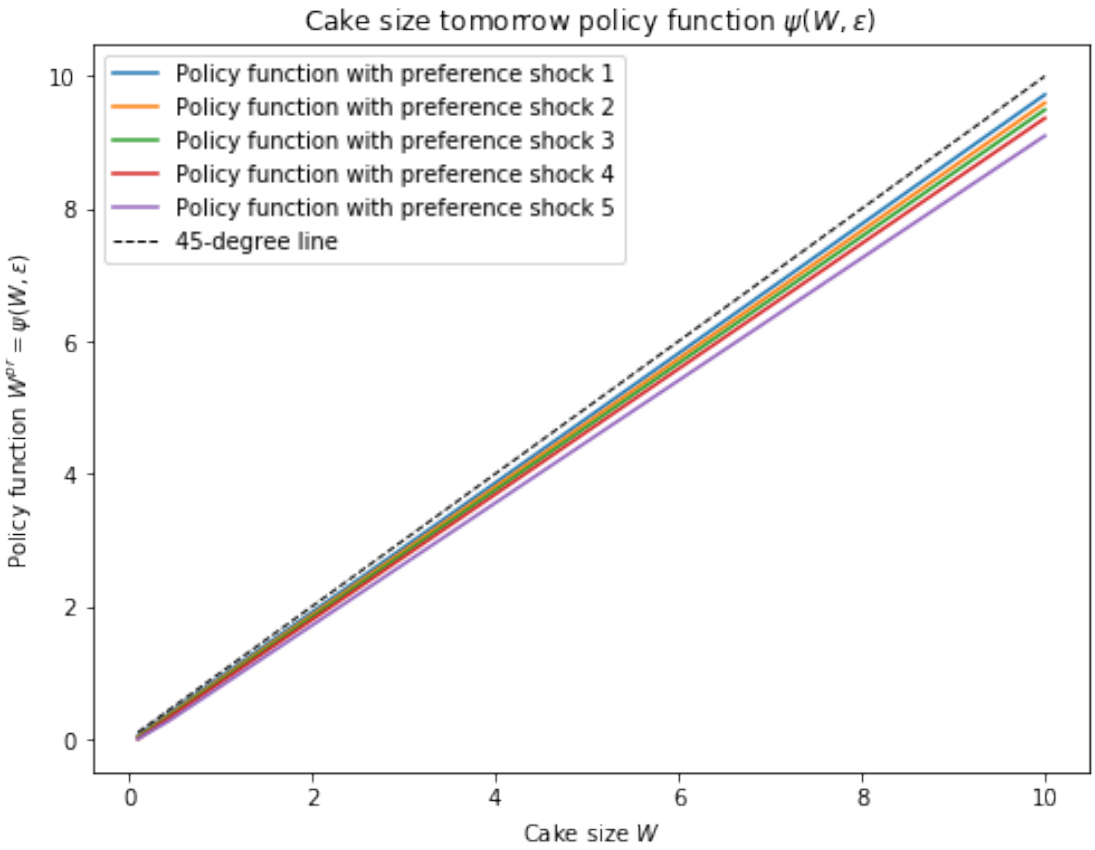
(b)

```
In [8]: plt.figure(figsize=(8,6))
        for i in range(5):
            plt.plot(W_vec, V_new[:,i], label='Value function with preference shock {}'.format
        plt.title('Equilibrium value function $V(W, \epsilon)$')
        plt.xlabel('Cake size $W$')
        plt.ylabel('Value function $V(W, \epsilon)$')
        plt.legend()

Out[8]: <matplotlib.legend.Legend at 0x110b046a0>
```



(c)

```
In [9]: plt.figure(figsize=(8,6))
        for i in range(5):
            plt.plot(W_vec, psi_mat[:,i], label='Policy function with preference shock {}'.forr
        plt.plot(W_vec, W_vec, color = 'black', linewidth=1, linestyle='--',
                 label='45-degree line')
        plt.title('Cake size tomorrow policy function $\psi(W, \epsilon)$')
        plt.xlabel('Cake size $W$')
        plt.ylabel('Policy function $W^{pr}=\psi(W, \epsilon)$')
        plt.legend()

Out[9]: <matplotlib.legend.Legend at 0x110c7aa20>
```


Cake size tomorrow policy function $\psi(W, \varepsilon)$
Policy function with preference shock 1
Policy function with preference shock 2
Policy function with preference shock 3
Policy function with preference shock 4
Policy function with preference shock 5
45-degree line
Policy function $W^{pr} = \psi(W, \varepsilon)$
Cake size W
0
2
4
6
8
10