526 Matlab Problem

Group L

Yuanyuan zhang, Xinye Xu, Yuxiang Wu

1. Summary

In Scenario 2, the maximum profit is ***173.88***. Full state should be set to ***6*** and Order Threshold state should be set to ***2***. Our average time of solving by Python is ***4.6998*** ms and standard deviation of that is ***1.2160*** ms (simulating 100 times). Firstly, we used MATLAB to implement algorithms. We found that finding transition distribution and solving linear equations for stationary distribution are two time-consuming processes in our algorithms. After that our improvement focused on these two parts. The design adapted Gaussian elimination instead of matrix inversion to solve stationary distribution. For finding transition matrix, we found a way to build matrix row by row and it can be wrote in 'list' format, then reshape it, preventing nested for loops. Because 0 ~ OT rows and F-th row will be the same, so we only need to consider 0T+1 ~ F-1 rows. Last but not the least, we guess that there is some relationship between the optimal policy and parameters (demand distribution, profit and cost per product). It is not necessary to go through all conditions ($5 \times 6 = 30$) if we can find it. The following picture is screenshot of output:

```
=============== senario 1 ===============
The maximum profit is 9.618181818181823
Full state should be set to 3 and Order Threshold state should be set to 1
=============== senario 2 ===============
The maximum profit is 173.88528720188106
Full state should be set to 6 and Order Threshold state should be set to 2
It took 0.004091739654541016 seconds.
```

Figure of output

2. *Code(by Python)*

```
# calculate transaction matrix
def tran_mat(F, OT, Dem):
    len_D = len(Dem)
    D = Dem + [0] * (F + 1 - len_D)    # when F > D, some remained stock prob=0
    D_r = D[::-1]
    mat_vec = D_r * (OT + 1)    # 0-OT rows have same prob
    for state in range(OT + 1, F):
        if state > len_D:
            row = [sum(Dem[state:])] * (state - (len_D - 1)) + Dem[:state][::-1]
        else:
            row = [sum(Dem[state:])] + Dem[:state][::-1]
        row = row + [0] * (F - state)
        mat_vec.extend(row)
    mat_vec.extend(D_r)    # full state-th row is the same as before
    return np.reshape(mat_vec, (F + 1, F + 1))

# solve stationary distribution
def stationary_d(P):
    P = P - np.eye(len(P))
    P[:,len(P)-1] = np.ones((1,len(P)))
    b = np.zeros((len(P),1))
    b[len(P)-1,:] = 1
    # use niave Gaussian elimination to solve linear equation
    a=np.transpose(P)
    return solve(a, b)

# Find optimal policy for inventory chain
def optimal(PPP,SCPP,F,OT,D):
```

```
    max_LRNPPD=0
    LRNPPD=np.zeros((2*len(F)+2,len(OT)))
    NPPD=np.zeros((2*len(F)+2,1))
    for i in F:
        for j in OT:
            P = tran_mat(i, j, D) # calculate transaction matrix
            Pi = stationary_d(P) # solve stationary distribution
            for k in range(0,i+1):
                #compute net profit per day under specific condition
                NPPD[k] = PPP*(i-k)*(k <= j)-SCPP*k
                # Compute long-run net profit per day
                LRNPPD[i,j]=LRNPPD[i,j]+Pi[k]*NPPD[k]
            if   LRNPPD[i,j] > max_LRNPPD: # FInd maximum profit
                max_LRNPPD=LRNPPD[i,j]
                optimal_F = i
                optimal_OT = j
    print('The maximum profit is',LRNPPD[optimal_F,optimal_OT])
    print('Full state shoud be set to',optimal_F,'and Order Threshold state shoud be set to',optimal_OT)


if __name__ == "__main__":

    PPP = 12 # profit per product
    SCPP = 2 # storage cost per product
    F = [3] # full state
    OT = [0, 1, 2] # order threshold
    D = [0.3, 0.4, 0.2, 0.1]# demand vector
    print("=============== senario 1 ===============")
    optimal(PPP,SCPP,F,OT,D)
```

```
start = time.time()

PPP = 100 # profit per product

SCPP = 15 # storage cost per product

F = [6, 7, 8, 9, 10] # full state

OT = [0, 1, 2, 3, 4, 5] # order threshold

D = [0.1, 0.2, 0.3, 0.25, 0.1, 0.05]# demand vector

print("=============== senario 2 ===============")

optimal(PPP,SCPP,F,OT,D)

print('It took', time.time()-start, 'seconds.')
```