Fixed Income HW5

Group Members: Huanyu Liu, Justin Tan, Tongsu Peng, Sajel Bharati

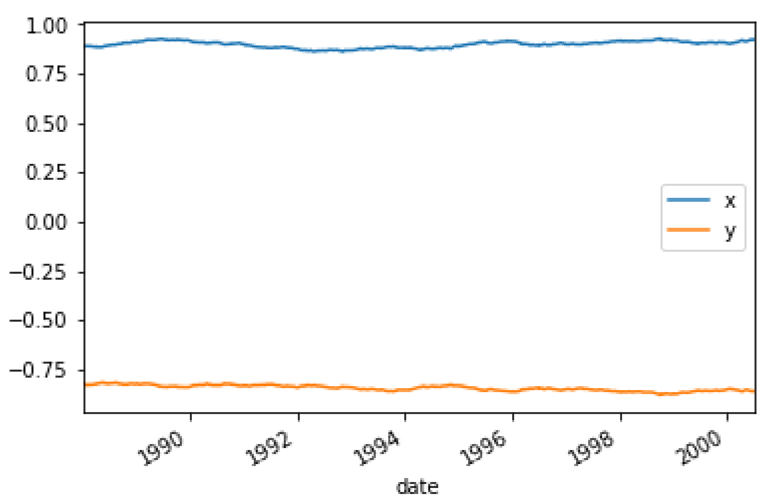
In [1]:
```
import pandas as pd
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

# 1-4

data = pd.read_excel('/Users/huanyu/Desktop/FixedIncome/hw5/Homework_5.x
lsx')
data[['cmt0.25','cmt2','cmt3','cmt5','cmt7','cmt10']] = data[['cmt0.25',
'cmt2','cmt3','cmt5','cmt7','cmt10']] / 100
data['date'] = np.vectorize(pd.datetime)(data['year'],data['month'],data
['day'])
alphax = 0.1
betax = 0.2
sigmax = 0.1
betay = 0.3
sigmay = 0.4

def A_T(alpha, beta, sigma, T):
    temp1 = (sigma ** 2 / (2 * beta * beta) - alpha / beta) * T
    temp2 = (alpha / (beta * beta) - sigma * sigma / beta ** 3) * (1 - n
p.exp(-beta * T))
    temp3 = sigma * sigma / (4 * beta ** 3) * (1 - np.exp(-2 * beta * T
))
    return np.exp(temp1 + temp2 + temp3)

def B_T(beta, T):
    return 1 / beta * (1 - np.exp(-beta * T))

def solve(x, y, a1, b1, c1, a2, b2, c2):
    a = np.array([[a1,b1],[a2,b2]])
    b = np.array([c1 + x, c2 + y])
    return np.linalg.solve(a, b)

def ytm(x,y,alphax,betax,sigmax,betay,sigmay,T):
    axT = A_T(alphax, betax, sigmax, T)
    ayT = A_T(0, betay, sigmay, T)
    bxT = B_T(betax, T)
    byT = B_T(betay, T)
    cxT = -np.log(axT) / T
    cyT = -np.log(ayT) / T
    return bxT / T * x + byT / T * y + cxT + cyT

def RMSE(parameters, data):
    alphax = parameters[0]
    betax = parameters[1]
    sigmax = parameters[2]
    betay = parameters[3]
    sigmay = parameters[4]
    if sigmax < 0 or sigmay < 0:
        return 10000
    ax0_25 = A_T(alphax, betax, sigmax, 0.25)
    bx0_25 = B_T(betax, 0.25)
    ay0_25 = A_T(0, betay, sigmay, 0.25)
    by0_25 = B_T(betay, 0.25)
```

```
    ax10 = A_T(alphax, betax, sigmax, 10)
    bx10 = B_T(betax, 10)
    ay10 = A_T(0, betay, sigmay, 10)
    by10 = B_T(betay, 10)
    a1 = bx0_25 / 0.25
    b1 = by0_25 / 0.25
    c1 = np.log(ax0_25) / 0.25 + np.log(ay0_25) / 0.25
    a2 = bx10 / 10
    b2 = by10 / 10
    c2 = np.log(ax10) / 10 + np.log(ay10) / 10
    x, y = solve(data['cmt0.25'], data['cmt10'], a1, b1, c1, a2, b2, c2)
    ytm2 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,2)
    ytm3 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,3)
    ytm5 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,5)
    ytm7 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,7)
    output = np.sum(np.square(data['cmt2'] - ytm2)) + np.sum(np.square(d
ata['cmt3'] - ytm3)) + np.sum(np.square(data['cmt5'] - ytm5)) + np.sum(n
p.square(data['cmt7'] - ytm7))
    return np.sqrt(output) / 4

RMSE([0.1,0.2,0.1,0.3,0.4],data)
output = opt.fmin(RMSE,x0=[0.1,0.2,0.1,0.3,0.4],args=(data,),maxiter=100
00)

alphax, betax, sigmax, betay, sigmay = output

ax0_25 = A_T(alphax,betax,sigmax,0.25)
bx0_25 = B_T(betax,0.25)
ay0_25 = A_T(0,betay,sigmay,0.25)
by0_25 = B_T(betay,0.25)
ax10 = A_T(alphax,betax,sigmax,10)
bx10 = B_T(betax,10)
ay10 = A_T(0,betay,sigmay,10)
by10 = B_T(betay,10)
a1 = bx0_25 / 0.25
b1 = by0_25 / 0.25
c1 = np.log(ax0_25) / 0.25 + np.log(ay0_25) / 0.25
a2 = bx10 / 10
b2 = by10 / 10
c2 = np.log(ax10) / 10 + np.log(ay10) / 10
x,y = solve(data['cmt0.25'],data['cmt10'], a1, b1, c1, a2, b2, c2)
data['x'] = x
data['y'] = y
print(data[['x','y']])
```

```
Optimization terminated successfully.
         Current function value: 0.025947
         Iterations: 846
         Function evaluations: 1352
            x         y
0    0.883269 -0.818148
1    0.888746 -0.823302
2    0.889688 -0.824689
3    0.891720 -0.827603
4    0.891046 -0.828973
5    0.888678 -0.826744
6    0.888974 -0.827023
7    0.890158 -0.828137
8    0.887991 -0.826599
9    0.888530 -0.826605
10   0.886753 -0.824933
11   0.886551 -0.824242
12   0.885071 -0.822849
13   0.888214 -0.824054
14   0.884614 -0.820917
15   0.885395 -0.820650
16   0.884050 -0.817881
17   0.886944 -0.818601
18   0.884374 -0.815681
19   0.887511 -0.816128
20   0.890822 -0.819744
21   0.890875 -0.820295
22   0.890081 -0.819047
23   0.893285 -0.821561
24   0.894469 -0.822675
25   0.892532 -0.819349
26   0.895373 -0.819517
27   0.896988 -0.819535
28   0.896840 -0.819395
29   0.894781 -0.818960
..        ...       ...
620  0.909300 -0.854865
621  0.907279 -0.852062
622  0.906474 -0.850704
623  0.903410 -0.849624
624  0.904708 -0.849142
625  0.902192 -0.846774
626  0.901398 -0.845526
627  0.904009 -0.847482
628  0.906346 -0.849380
629  0.904453 -0.847900
630  0.906526 -0.849850
631  0.909274 -0.851835
632  0.908418 -0.851330
633  0.908968 -0.851447
634  0.913016 -0.854354
635  0.913016 -0.854354
636  0.916769 -0.856984
637  0.919423 -0.859381
638  0.919296 -0.859462
639  0.917498 -0.857570
640  0.913808 -0.854198
```

```
641  0.913233 -0.851052
642  0.915293 -0.851487
643  0.915293 -0.851487
644  0.917546 -0.853907
645  0.921309 -0.856647
646  0.921056 -0.856809
647  0.922991 -0.858730
648  0.920168 -0.855974
649  0.922674 -0.858231

[650 rows x 2 columns]
```

In [2]:
```
# 5

data[['date','x','y']].plot(x='date')
plt.show()

x_mean = data['x'].mean()
x_std = data['x'].std(ddof = 1)
y_mean = data['y'].mean()
y_std = data['y'].std(ddof = 1)
Ex = alphax / betax
Vx = sigmax * sigmax / (2 * betax)
diff_expectation = abs(x_mean - Ex)
diff_volatility = abs(x_std - Vx)
print(x_mean)
print(x_std)
print(Ex)
print(Vx)
```

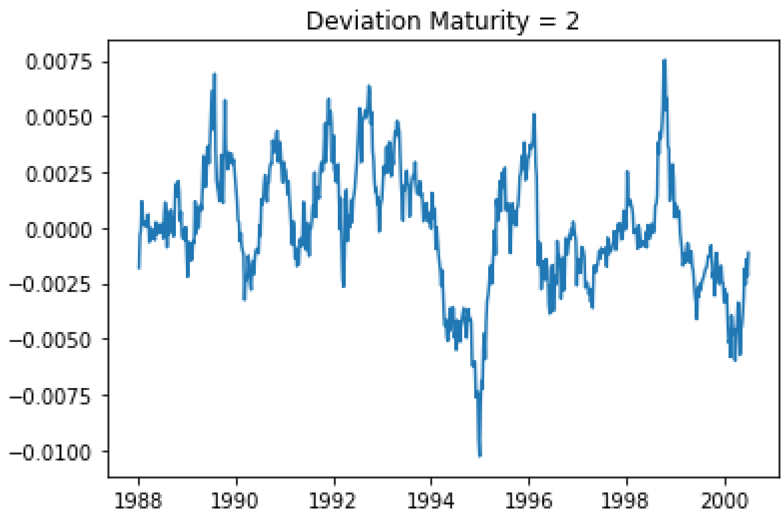


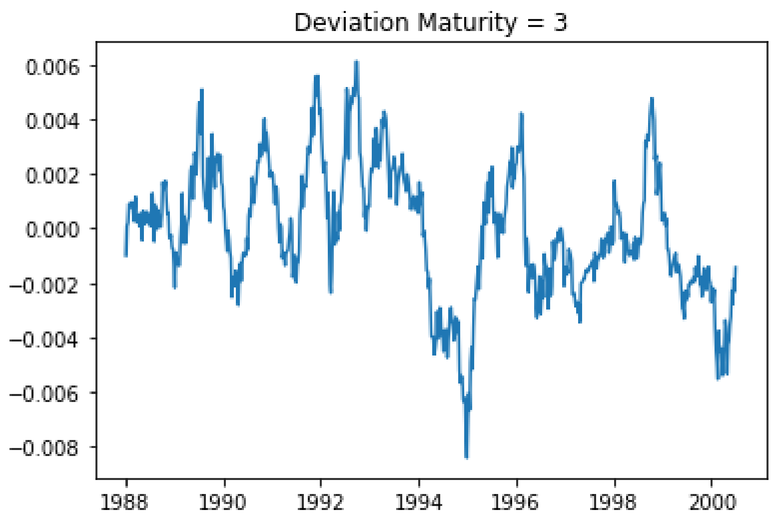

```
0.8980985373657558
0.016730662976453165
0.8987446544006794
0.014494070547204777
```

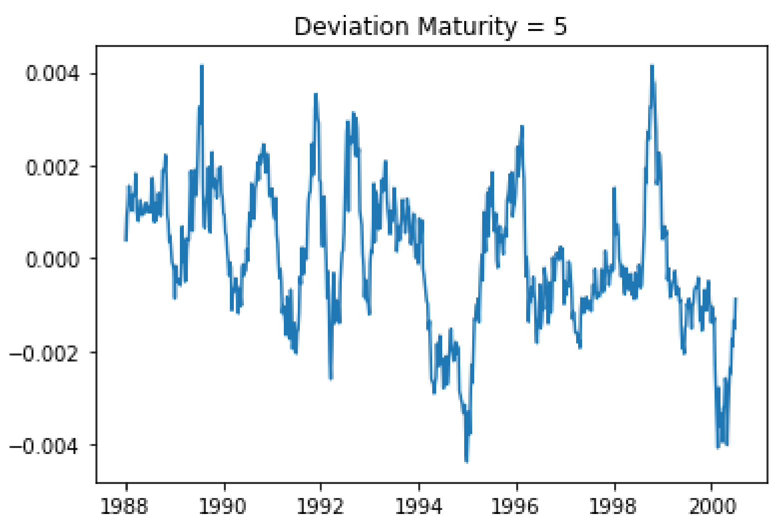
Sample means of X is 0.8981, and implied mean of X is 0.8987.
Sample volatility of X is 0.01673, and implied volatility of X is 0.01449.

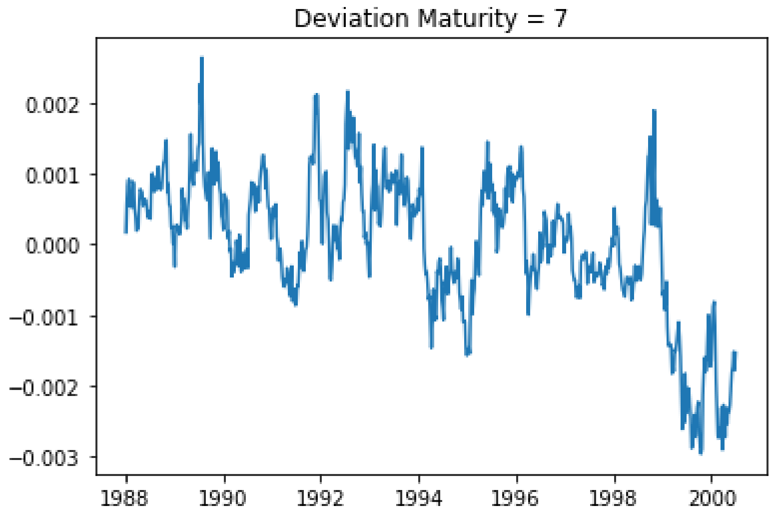
In [3]:

```
# 6

ytm2 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,2)
ytm3 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,3)
ytm5 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,5)
ytm7 = ytm(x,y,alphax,betax,sigmax,betay,sigmay,7)
deviation2 = ytm2 - data['cmt2']
deviation3 = ytm3 - data['cmt3']
deviation5 = ytm5 - data['cmt5']
deviation7 = ytm7 - data['cmt7']

plt.plot(data['date'],deviation2)
plt.title('Deviation Maturity = 2')
plt.show()
plt.plot(data['date'],deviation3)
plt.title('Deviation Maturity = 3')
plt.show()
plt.plot(data['date'],deviation5)
plt.title('Deviation Maturity = 5')
plt.show()
plt.plot(data['date'],deviation7)
plt.title('Deviation Maturity = 7')
plt.show()
```


Deviation Maturity = 2
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100
1988
1990
1992
1994
1996
1998
2000


Deviation Maturity = 3
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
1988
1990
1992
1994
1996
1998
2000


Deviation Maturity = 5
0.004
0.002
0.000
−0.002
−0.004
1988
1990
1992
1994
1996
1998
2000


Deviation Maturity = 7
0.002
0.001
0.000
−0.001
−0.002
−0.003
1988
1990
1992
1994
1996
1998
2000

In [4]:

```
# 7

def duration(rate,T):
    face_value = 100
    coupon = face_value * rate / 2
    period = T * 2
    dur = 0
    for i in range(1,period+1):
        dur += coupon * np.exp(-i * rate/2) / face_value * i * 0.5
    dur += np.exp(-T * rate) * T
    return dur * np.exp(-rate/2)

dur2 = duration(0.0792,2)
dur5 = duration(0.0848,5)
dur10 = duration(0.0897,10)
def convexity(rate,T):
    rate_up = rate + 0.0001
    rate_down = rate - 0.0001
    dur_up = duration(rate_up,T)
    dur_down = duration(rate_down,T)
    convex = (dur_down - dur_up) / (rate_up - rate_down)
    return convex

convex2 = convexity(0.0792,2)
convex5 = convexity(0.0848,5)
convex10 = convexity(0.0897,10)

N2a, N10a = np.linalg.solve([[dur2,dur10],[convex2,convex10]],[-dur5,-co
nvex5])
print(N2a)
print(N10a)
def D_T(x, y, alphax, betax, sigmax, betay, sigmay, T):
    axT = A_T(alphax, betax, sigmax, T)
    ayT = A_T(0, betay, sigmay, T)
    bxT = B_T(betax, T)
    byT = B_T(betay, T)
    return axT * ayT * np.exp(-bxT * x - byT * y)

def derivative(rate,alphax, betax, sigmax, betay, sigmay,beta,T):
    coupon = 100 * rate / 2
    sum = 0
    for i in range(1,2 * T + 1):
        dt = D_T(0.88327,-0.81815,alphax, betax, sigmax, betay, sigmay,i
/2)
        sum += B_T(beta,i/2) * dt
    output = -coupon * sum - D_T(.88327,-0.81815,alphax, betax, sigmax,
betay, sigmay,T) * B_T(beta,T)
    return output

dx2 = derivative(0.0792,alphax, betax, sigmax, betay, sigmay,betax,2)
dx5 = derivative(0.0848,alphax, betax, sigmax, betay, sigmay,betax,5)
dx10 = derivative(0.0897,alphax, betax, sigmax, betay, sigmay,betax,10)
dy2 = derivative(0.0792,alphax, betax, sigmax, betay, sigmay,betay,2)
dy5 = derivative(0.0848,alphax, betax, sigmax, betay, sigmay,betay,5)
dy10 = derivative(0.0897,alphax, betax, sigmax, betay, sigmay,betay,10)
N2b, N10b = np.linalg.solve([[dx2,dx10],[dy2,dy10]],[-dx5,-dy5])
```

```python
print(N2b)
print(N10b)
```

```
-1.2930211140274779
-0.25495194204473026
-2.01430028814361
-0.19556100646908053
```

Duration and convexity approach: N2 = -1.2930, N10 = -0.2550.
Derivative approach: N2 = -2.0143, N10 = -0.1956.