In [1]:
```
import pandas as pd
import numpy as np
import quandl
from sklearn.decomposition import PCA
```

2a. Download a panel of CMT rates into pandas data frame & remove '1M column from the dataset

In [3]:
```
# 1. Definition
sample_start = '2012-09-30'
sample_end = '2016-09-30'
```

In [4]:
```
# 2.a
df = quandl.get("USTREASURY/YIELD", authtoken="z7H963w1dPBFCq_p2Eh6")
df = df.drop(columns = ['1 MO','2 MO'], errors = 'ignore') # since 2 Mo also shows nan, I delete 2 Mo as well
df = df.dropna()

#compute a table with daily changes:
# we examine the head and tail to make sure everything looks okay
df_ret = df.diff(periods=1).dropna(axis = 0)
print (df_ret.head(2))
print (df_ret.tail(2))
```

```
            3 MO  6 MO  1 YR  2 YR  3 YR  5 YR  7 YR  10 YR  20 YR  30 YR
Date
1993-10-04  0.04  0.06  0.00  0.01 -0.02 -0.01  0.01   0.00  -0.02   0.01
1993-10-05  0.04  0.03  0.03  0.01  0.02  0.01  0.01   0.01   0.02   0.02
            3 MO  6 MO  1 YR  2 YR  3 YR  5 YR  7 YR  10 YR  20 YR  30 YR
Date
2023-04-07  0.04  0.02  0.10  0.15  0.13  0.12  0.11   0.09   0.07   0.07
2023-04-10  0.13  0.03  0.04  0.03  0.03  0.03  0.02   0.02   0.01   0.01
```

b. Perform PCA on the dataset using Sample1

In [5]:
```
sample_ret = df_ret[(df_ret.index>=sample_start) & (df_ret.index<=sample_end)]
print(sample_ret.head(5))
print(sample_ret.tail(5))

# 1 factor pca model
pca = PCA(n_components=10)
pca.fit(sample_ret)
val = pca.explained_variance_
vec = pca.components_
```

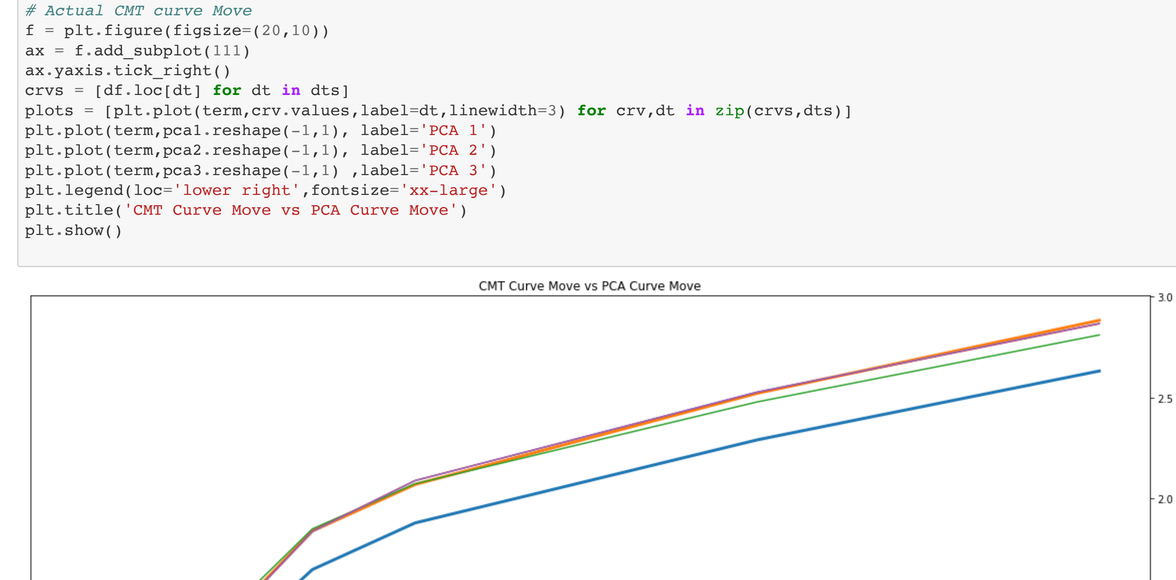
```
            3 MO  6 MO  1 YR  2 YR  3 YR  5 YR  7 YR  10 YR  20 YR  30 YR
Date
2012-10-01 -0.01  0.00  0.00  0.02  0.00  0.00  0.00  -0.01  -0.01  -0.01
2012-10-02  0.00  0.00 -0.01 -0.02  0.00 -0.01 -0.01   0.00   0.00   0.00
2012-10-03  0.00  0.00  0.00  0.00  0.00  0.00 -0.01   0.00   0.01   0.01
2012-10-04  0.01  0.00  0.02  0.00  0.01  0.02  0.05   0.06   0.06   0.07
2012-10-05  0.01  0.01  0.00  0.04  0.02  0.04  0.05   0.05   0.07   0.07
            3 MO  6 MO  1 YR  2 YR  3 YR  5 YR  7 YR  10 YR  20 YR  30 YR
Date
2016-09-26  0.07  0.02 -0.02 -0.01 -0.03 -0.03 -0.03  -0.03  -0.02  -0.02
2016-09-27  0.01  0.00  0.00 -0.01 -0.01 -0.01 -0.02  -0.03  -0.04  -0.04
2016-09-28  0.01  0.02  0.02  0.00  0.01  0.01  0.02   0.01   0.00   0.01
2016-09-29 -0.01 -0.01 -0.01 -0.02 -0.02 -0.01 -0.02  -0.01  -0.01  -0.01
2016-09-30  0.03  0.02  0.00  0.04  0.03  0.02  0.03   0.04   0.04   0.04
```

c. Use this PCA model to analyze the CMT curve move on the 2016 Election Day: 11/8/2016 to 11/9/2016

In [6]:
```
#Election Day: 11/8/2016 to 11/9/2016
factor_matrix=df_ret['2016-11-09':'2016-11-09']@vec[0:1].T
delta_y1=factor_matrix@vec[0:1]
pca1 = df['2016-11-08':'2016-11-08'].values+delta_y1.values

# 2 factors pca model
factor_matrix=df_ret['2016-11-09':'2016-11-09']@vec[0:2].T
delta_y2=factor_matrix@vec[0:2]
pca2 = df['2016-11-08':'2016-11-08'].values+delta_y2.values

# 3 factors pca model

factor_matrix=df_ret['2016-11-09':'2016-11-09']@vec[0:3].T
delta_y3=factor_matrix@vec[0:3]
pca3 = df['2016-11-08':'2016-11-08'].values+delta_y3.values
```

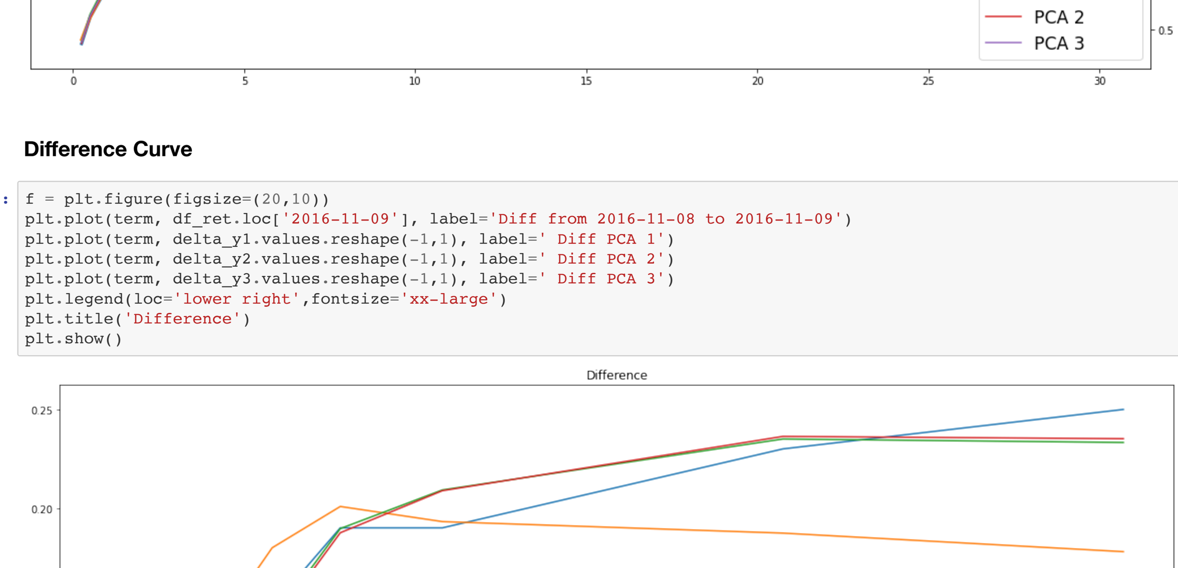
c.i. Plot CMT curve move vs the move explained by the first PCA factor, first 2 PCA factors, first 3 PCA factors

In [7]:
```
import matplotlib.pyplot as plt

term = [3/12,6/12,1,2,3,5,7,10,20,30]
dts = ['2016-11-08', '2016-11-09']

# Actual CMT curve Move
f = plt.figure(figsize=(20,10))
ax = f.add_subplot(111)
ax.yaxis.tick_right()
crvs = [df.loc[dt] for dt in dts]
plots = [plt.plot(term,crv.values,label=dt,linewidth=3) for crv,dt in zip(crvs,dts)]
plt.plot(term,pca1.reshape(-1,1), label='PCA 1')
plt.plot(term,pca2.reshape(-1,1), label='PCA 2')
plt.plot(term,pca3.reshape(-1,1) ,label='PCA 3')
plt.legend(loc='lower right',fontsize='xx-large')
plt.title('CMT Curve Move vs PCA Curve Move')
plt.show()
```



Difference Curve

In [8]:
```
f = plt.figure(figsize=(20,10))
plt.plot(term, df_ret.loc['2016-11-09'], label='Diff from 2016-11-08 to 2016-11-09')
plt.plot(term, delta_y1.values.reshape(-1,1), label=' Diff PCA 1')
plt.plot(term, delta_y2.values.reshape(-1,1), label=' Diff PCA 2')
plt.plot(term, delta_y3.values.reshape(-1,1), label=' Diff PCA 3')
plt.legend(loc='lower right',fontsize='xx-large')
plt.title('Difference')
plt.show()
```



c(ii. Explain your calculations and results

In [11]:
```
print('move explained by PCA1')
print(pca1)
print('move explained by PCA2')
print(pca2)
print('move explained by PCA3')
print(pca3)
```

```
move explained by PCA1
[[0.43836787 0.57504508 0.73909818 0.96022368 1.17018409 1.51989679
  1.85083354 2.07316174 2.47733453 2.80792828]]
move explained by PCA2
[[0.43115024 0.56066915 0.71800498 0.90768073 1.11395204 1.48278157
  1.83960753 2.08920882 2.52503272 2.86327562]]
move explained by PCA3
[[0.43713799 0.56733547 0.72275367 0.90889944 1.11337074 1.48047952
  1.83751338 2.08883992 2.52634249 2.86517381]]
```

In [13]:
```
print(np.around(pca.explained_variance_ratio_*100, decimals = 2).reshape(10,1)) # Explained variance ratio (%)
```

```
[[85.48]
 [ 7.77]
 [ 2.39]
 [ 1.39]
 [ 1.01]
 [ 0.65]
 [ 0.56]
 [ 0.3 ]
 [ 0.24]
 [ 0.22]]
```

1. From PCA method above, we know that the first pca factor could explain 85.48% change of the CMT rates, while the second factor and the third factor's contributions are 7.7% and 2.39% respectively. And the real results could prove that.
2. As we cam see from the above graphs, the move PCA 1 performs is not close to the real curve so I would say it doesn't explain very well, but the curves PCA 2 and PCA 3 performs is close to the real curve, meaning that they explain much better than PCA 1. Thus, the more PCA factors we select, the more explained by PCA.

d. Compute weights of the WFLY to make sure that WFLY does not have PCA1,2 risk exposure in Sample1. Let's call this combination WFLY1

wfly= W1x3Y - 5Y + w2x7Y weight = (w1, -1, w2)

In [15]:
```
sample_level = df[(df.index>=sample_start) & (df.index<=sample_end)]
sample_level.head()
```

Out[15]:

	3 MO	6 MO	1 YR	2 YR	3 YR	5 YR	7 YR	10 YR	20 YR	30 YR
Date										
2016-09-26	0.25	0.42	0.58	0.76	0.87	1.13	1.41	1.59	2.00	2.32
2016-09-27	0.26	0.42	0.58	0.75	0.86	1.12	1.39	1.56	1.96	2.28
2016-09-28	0.27	0.44	0.60	0.75	0.87	1.13	1.41	1.57	1.96	2.29
2016-09-29	0.26	0.43	0.59	0.73	0.85	1.12	1.39	1.56	1.95	2.28
2016-09-30	0.29	0.45	0.59	0.77	0.88	1.14	1.42	1.60	1.99	2.32

In [16]:
```
# Extract components of 3y(4), 5y(5), and 7y(6)

a = vec[0,4]
b = vec[0,6]

c = vec[1,4]
d = vec[1,6]

e = vec[0,5]
f = vec[1,5]

WFLY1_w = np.linalg.solve(np.array([[a,b],[c,d]]),np.array([e,f]))

WFLY1_w =[WFLY1_w[0], -1, WFLY1_w[1]]
print('weights of the WFLY1:\n [3Yr,  5Yr,  7Yr]:', WFLY1_w)
```

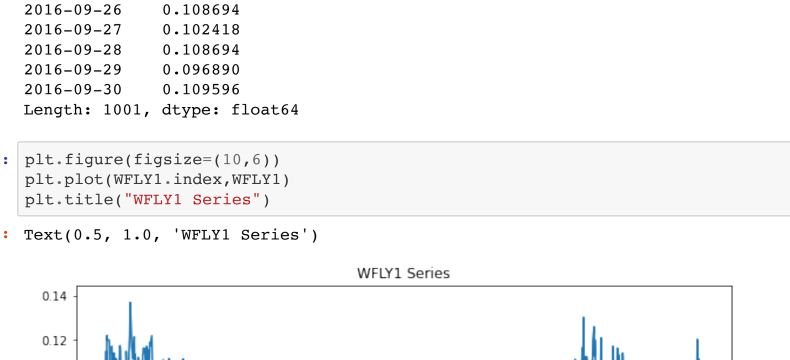
```
weights of the WFLY1:
 [3Yr,  5Yr,  7Yr]: [0.5527410405025863, -1, 0.537453574304419]
```

In [18]:
```
WFLY1 = WFLY1_w[0]*sample_level['3 YR'] + WFLY1_w[1]*sample_level['5 YR'] + WFLY1_w[2]*sample_level['7 YR']
WFLY1
```

Out[18]:

```
Date
2012-10-01    0.110301
2012-10-02    0.114927
```

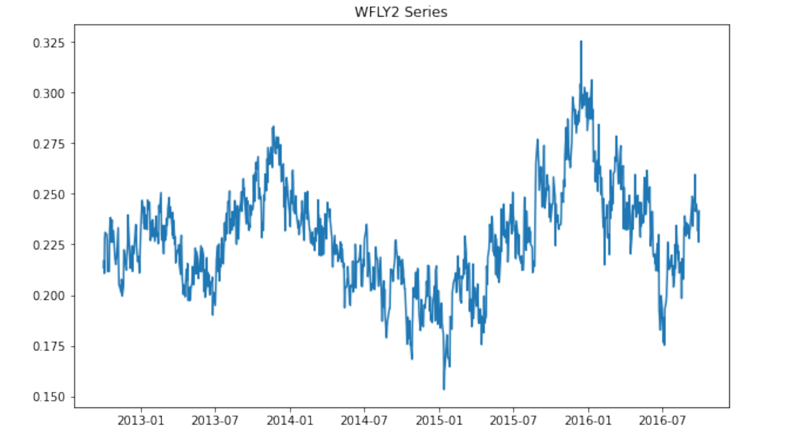
[illegible]



e. Choose weights of the WFLY from cointegration analysis (weights correspond to the best cointegrated vector). Let's call this combination WFLY2

i. Use Box-Tiao estimation procedure

[illegible]



e(ii. If cointegration estimation fails for you – use a linear regression of levels instead (regressing 5Y rate on [3Y,7Y] rate)

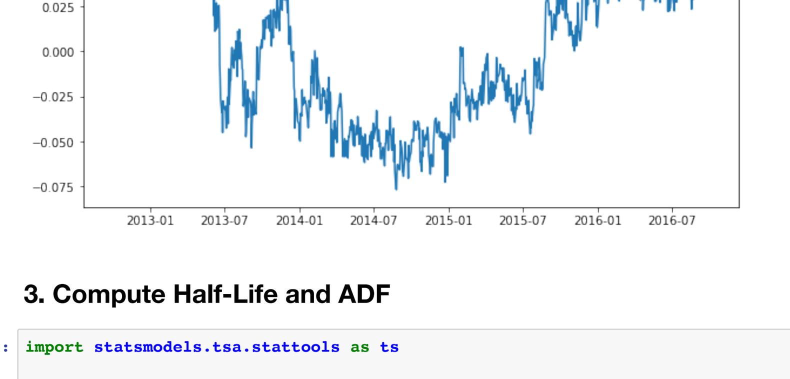
In [26]:
```
x=sample_level[['3 YR','7 YR']]
y=sample_level[['5 YR']]

reg = LinearRegression(fit_intercept=False)
reg.fit(x,y)
c = reg.coef_
LR_w=[c[0,0],-1,c[0,1]]
print('weights of WFLY2 using Linear Regression:\n [3Yr,  5YR, 7Yr]:', LR_w)
```

```
weights of WFLY2 using Linear Regression:
 [3Yr,  5YR, 7Yr]: [0.49911280303613836, -1, 0.5356698561551125]
```

In [27]:
```
# Time Series from Linear Regression
WFLY_LR=LR_w[0]*sample_level['3 YR'] + LR_w[1]*sample_level['5 YR'] + LR_w[2]*sample_level['7 YR']
```

In [34]:
```
plt.figure(figsize=(10,6))
plt.plot(WFLY_LR.index,WFLY_LR)
plt.title("WFLY_LR Series")
```

Out[34]: `Text(0.5, 1.0, 'WFLY_LR Series')`



3. Compute Half-Life and ADF

In [48]:
```
import statsmodels.tsa.stattools as ts

def half_life(data):
    x=data.shift().dropna().values.reshape(-1,1)
    y=data.diff().dropna().values.reshape(-1,1)
    reg = LinearRegression()
    reg.fit(x,y)

    half_life = -np.log(2)/reg.coef_[0]
    halflife = np.round(half_life,1)
    print("The half life of time series is:", np.around(float(half_life),decimals = 3))

def adf(data):
    adf = ts.adfuller(data)
    print('ADF statistics:\n',adf,'\n')

    if adf[0]<=adf[4]['5%']:
        print('Since ADF statistics is smaller than t-statistics at 95% confidence level.')
        print('The Time Series is stationary\n')
    else:
        print('Since ADF Statistics is greater than t-statistics at 95% confidence level.')
        print('The Time series is not stationary\n')
```

In [51]:
```
# WFLY2 via Box Tiao
half_life(WFLY2)
adf(WFLY2)

# WFLY2 via LR
half_life(WFLY_LR)
adf(WFLY_LR)
```

```
The half life of time series is: 10.499
ADF statistics:
 (-3.512486910079349, 0.00767213546528639, 3, 997, {'1%': -3.4369259442540416, '5%': -2.8644432969122833, '10%': -2.5
683158550174094}, -6488.939002448372)

Since ADF statistics is smaller than t-statistics at 95% confidence level.
The Time Series is stationary

The half life of time series is: 41.741
ADF statistics:
 (-2.1629517893555894, 0.21996543889878728, 3, 997, {'1%': -3.4369259442540416, '5%': -2.8644432969122833, '10%': -2.
5683158550174094}, -6703.172600749939)

Since ADF Statistics is greater than t-statistics at 95% confidence level.
The Time series is not stationary
```

4. 1. Repeat Step #3 out-of-sample: using 3m, 6m out of sample periods

In [54]:
```
# out of sample
out_level_3m = df.loc['2016-10-01':'2017-01-01', :]
out_level_6m = df.loc['2016-10-01':'2017-04-01', :]
```

In [56]:
```
# Use Box-Tiao estimation procedure
WFLY2_3m = WFLY2_w[0]*out_level_3m.loc[:,'3 YR'] + WFLY2_w[1]*out_level_3m.loc[:,'5 YR'] + WFLY2_w[2]*out_level_3m.loc
[:,'7 YR']
WFLY2_6m = WFLY2_w[0]*out_level_6m.loc[:,'3 YR'] + WFLY2_w[1]*out_level_6m.loc[:,'5 YR'] + WFLY2_w[2]*out_level_6m.loc
[:,'7 YR']
```

In [57]:
```
# Compute Half-Life & ADF statistic for 3m out of sample periods
half_life(WFLY2_3m)
adf(WFLY2_3m)

half_life(WFLY2_6m)
adf(WFLY2_6m)
```

```
The half life of time series is: 6.748
ADF statistics:
 (-1.518603011923469, 0.5242894466824914, 1, 59, {'1%': -3.5463945337644063, '5%': -2.911939409384601, '10%': -2.5936
515282964665}, -315.88324143151306)

Since ADF Statistics is greater than t-statistics at 95% confidence level.
The Time series is not stationary

The half life of time series is: 5.156
ADF statistics:
 (-2.487809164445346, 0.11846873302910715, 1, 121, {'1%': -3.485585145896754, '5%': -2.885738566292665, '10%': -2.579
6759080663887}, -713.682110831519)

Since ADF Statistics is greater than t-statistics at 95% confidence level.
The Time series is not stationary
```

In [65]:
```
# Use linear regression estimation procedure
WFLY_LR_3m = LR_w[0]*out_level_3m.loc[:,'3 YR'] + LR_w[1]*out_level_3m.loc[:,'5 YR'] + LR_w[2]*out_level_3m.loc[:,'7 Y
R']
WFLY_LR_6m = LR_w[0]*out_level_6m.loc[:,'3 YR'] + LR_w[1]*out_level_6m.loc[:,'5 YR'] + LR_w[2]*out_level_6m.loc[:,'7 Y
R']
```

In [66]:
```
#linear regression
# Compute Half-Life & ADF statistic for 3m out of sample periods
print('Linear Regression Method:\n')
half_life(WFLY_LR_3m)
adf(WFLY_LR_3m)

# Compute Half-Life & ADF statistic for 6m out of sample periods
half_life(WFLY_LR_6m)
adf(WFLY_LR_6m)
```

```
Linear Regression Method:

The half life of time series is: 10.132
ADF statistics:
 (-0.9509541401511369, 0.770729975257124, 1, 59, {'1%': -3.5463945337644063, '5%': -2.911939409384601, '10%': -2.5936
515282964665}, -319.6190276967109)

Since ADF Statistics is greater than t-statistics at 95% confidence level.
The Time series is not stationary

The half life of time series is: 8.321
ADF statistics:
 (-1.84045351156153, 0.36057053347572887, 1, 121, {'1%': -3.485585145896754, '5%': -2.885738566292665, '10%': -2.5796
759080663887}, -745.4518380407154)

Since ADF Statistics is greater than t-statistics at 95% confidence level.
The Time series is not stationary
```

a. How do out-of-sample results compare across periods and combinations?

All of the out of sample time series, no matter what their weights are or how long the out-of-sample periods are, they are all non-stationary.The variability in economic conditions over time may affect the long-run mean; for instance, the change in the monetary policy stance and market perception of Fed's policy action may have an impact on the long-run rates, causing non-stationarity as evidenced across different time periods in the series.

Besides, we find that WFLY2 via Box Taio Method is stationary in-sample, so we can say CCA by Box-Taio shows its ability of identifying mean reverting portfolio at least in-sample. However, the Linear Regression shows both non-stationary for the both in and out of sample, meaning that it may not a good alternative method if cointegration estimation fails for us.

In [ ]: