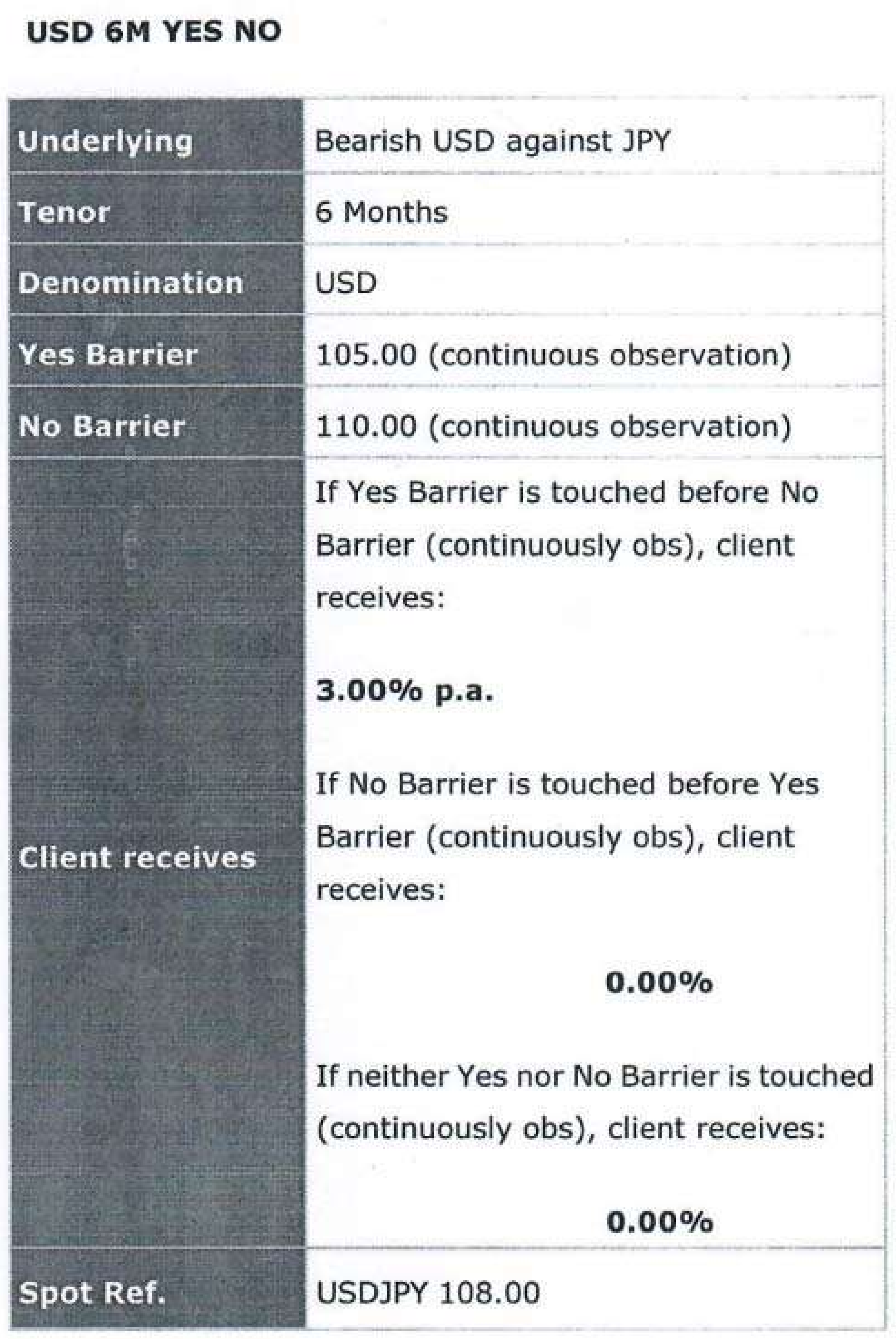
財務演算法作業二

台大財金所財工組碩一 周永昱 **R08723058**

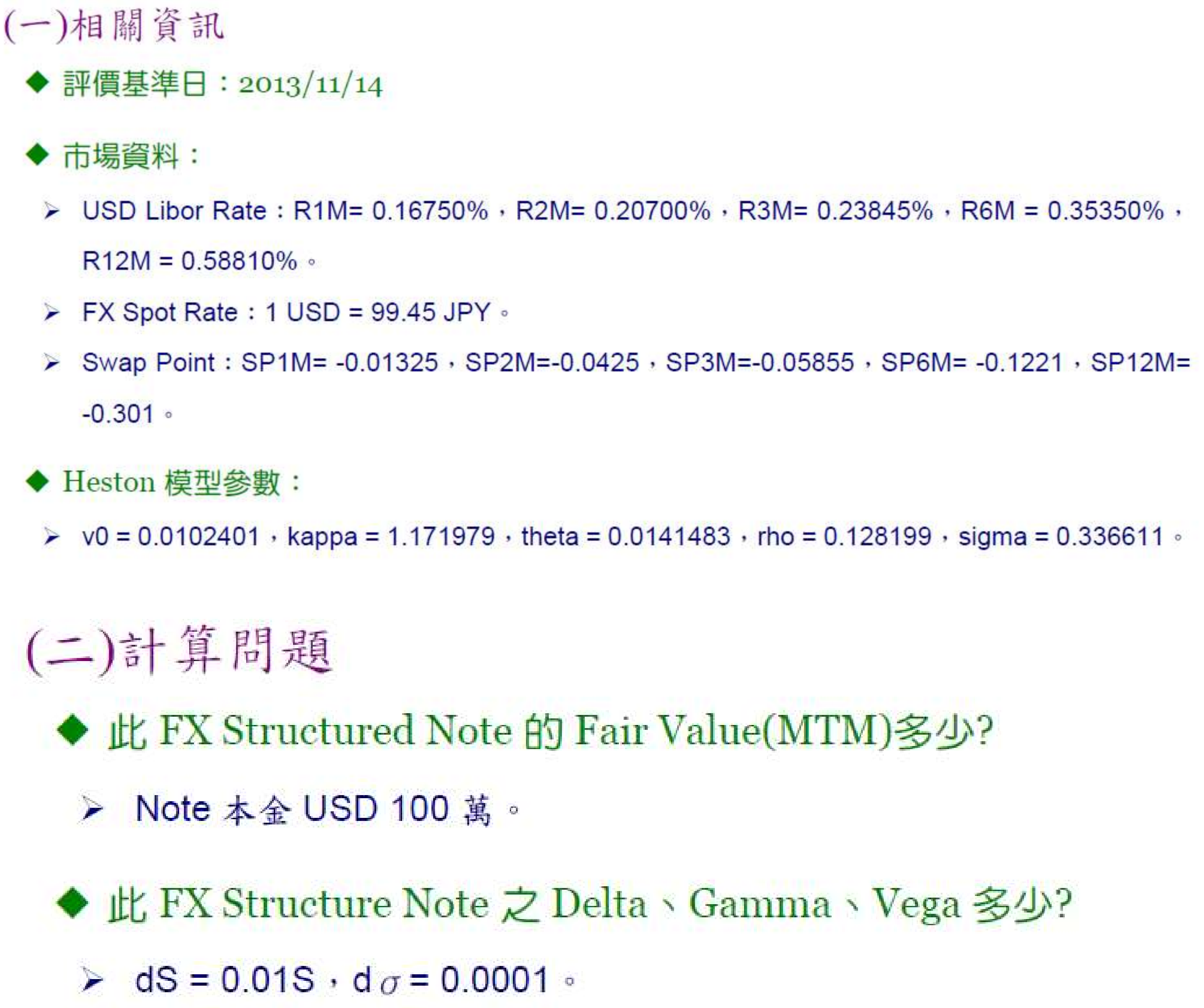
目錄

一、作業說明



USD 6M YES NO

Underlying	Bearish USD against JPY
Tenor	6 Months
Denomination	USD
Yes Barrier	105.00 (continuous observation)
No Barrier	110.00 (continuous observation)
Client receives	If Yes Barrier is touched before No Barrier (continuously obs), client receives: **3.00% p.a.** If No Barrier is touched before Yes Barrier (continuously obs), client receives: **0.00%** If neither Yes nor No Barrier is touched (continuously obs), client receives: **0.00%**
Spot Ref.	USDJPY 108.00



(一)相關資訊

- 評價基準日：2013/11/14
- 市場資料：
  - USD Libor Rate：R1M= 0.16750%，R2M= 0.20700%，R3M= 0.23845%，R6M = 0.35350%，R12M = 0.58810%。
  - FX Spot Rate：1 USD = 99.45 JPY。
  - Swap Point：SP1M= -0.01325，SP2M=-0.0425，SP3M=-0.05855，SP6M= -0.1221，SP12M= -0.301。
- Heston 模型參數：
  - v0 = 0.0102401，kappa = 1.171979，theta = 0.0141483，rho = 0.128199，sigma = 0.336611。

(二)計算問題

- 此 FX Structured Note 的 Fair Value(MTM)多少?
  - Note 本金 USD 100 萬。
- 此 FX Structure Note 之 Delta、Gamma、Vega 多少?
  - dS = 0.01S，dσ = 0.0001。

注意事項

FX spot Rate + Swap Point -> 遠期匯率

summarize重要結論
重要資訊: 姓名學號系級
寄到dongmy@msa.hinet.net

二、參數設置

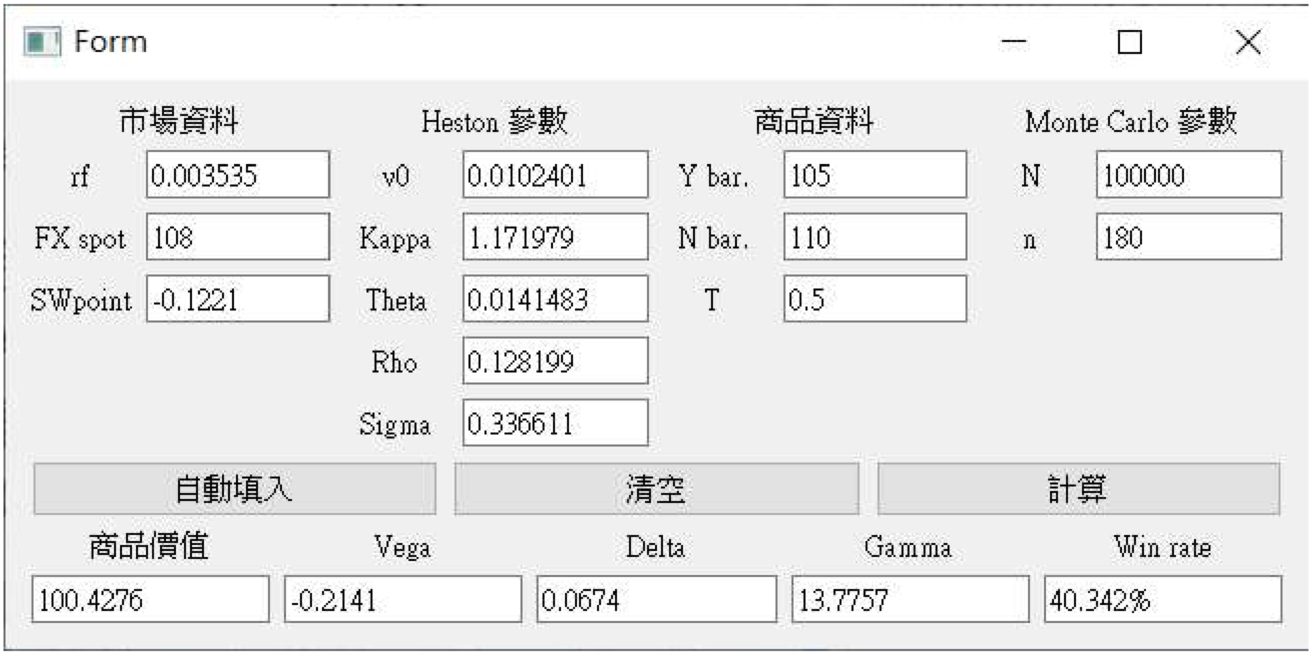
In [1]:

```
%matplotlib inline
import numpy as np
from math import sqrt, exp, log
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

In [58]:

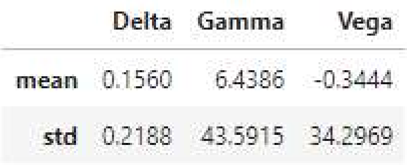
```
## 市場資料
T = 1/2
rf1m, rf2m, rf3m, rf6m, rf12m = 0.001675, 0.00207,0.0023845,0.0035350,0.005881
fx_spot = 108
sp1m, sp2m, sp3m, sp6m, sp12m = -0.01325, -0.0425, -0.05855, -0.1221, -0.301
y_barrier = 105
n_barrier = 110

fx_6m = fx_spot + sp6m # 6個月的遠期匯率
r6m = ((1+rf6m/2)*fx_6m/fx_spot - 1)*2
## Heston Model 參數
## 應該是要用交易的資料calibration，但這邊老師直接給

v0 = 0.0102401
kappa = 1.171979
theta = 0.0141483
rho = 0.128199
sigma = 0.336611

n = 180
nos = 100000 # number of simulation
xright = n+5
```

三、計算結果



市場資料		Heston 參數		商品資料		Monte Carlo 參數	
rf	0.003535	v0	0.0102401	Y bar.	105	N	100000
FX spot	108	Kappa	1.171979	N bar.	110	n	180
SWpoint	-0.1221	Theta	0.0141483	T	0.5		
		Rho	0.128199				
		Sigma	0.336611				

商品價值	Vega	Delta	Gamma	Win rate
100.4276	-0.2141	0.0674	13.7757	40.342%



	Delta	Gamma	Vega
mean	0.1560	6.4386	-0.3444
std	0.2188	43.5915	34.2969

四、單條路徑模擬及模擬結果作圖

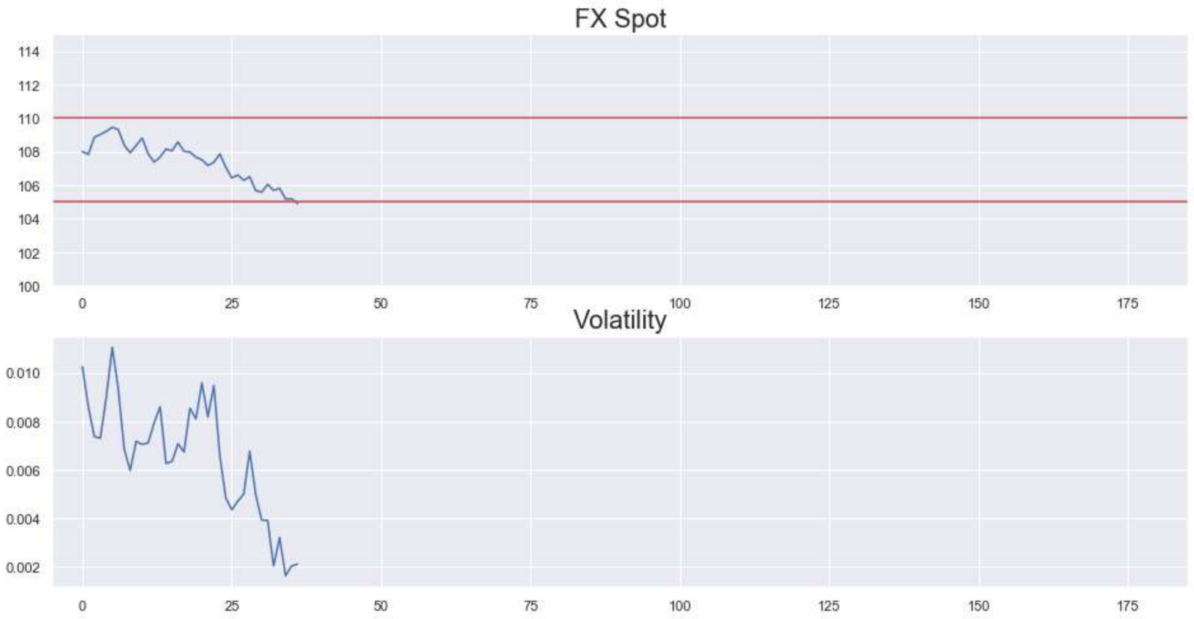
In [3]:

```
import numpy as np
from math import sqrt, exp, log

# 跑一次monte_carlo得一條路徑，看使否得到報酬
def monte_carlo(r,fx_spot,y_barrier,n_barrier,rf,v0,kappa,theta,rho,sigma,n,T,seed=None):

    ## 如果有指定隨機種子則用指定的，否則隨機指派本次模擬的隨機種子
    if seed == None:
        seed = np.random.randint(100000)
    np.random.seed = seed

    ## 抽樣
    zero_cov_rand = np.random.normal(size=(n,2)) #抽零相關的樣本
    cov_matrix = np.array([[1,rho],[rho,1]]) #共變異數矩陣
    cho_cov = np.linalg.cholesky(cov_matrix) # cholesky
    rho_cov_rand = zero_cov_rand.dot(cho_cov.T) #有相關性的brownian motion

    ## 模擬 S, V路徑，檢查是否得到報酬
    S = [fx_spot]
    V = [v0]
    dt = T/n
    touch_Y = False
    touch_N = False
    get_payoff = False
    for i in range(n):
        dS = (r-rf)*S[-1]*dt + sqrt(V[-1])*S[-1]*rho_cov_rand[i][0]*sqrt(dt)
        dV = kappa*(theta-V[-1])*dt + sigma*sqrt(V[-1])*rho_cov_rand[i][1]*sqrt(dt)
        V.append(abs(V[-1]+dV))
        S.append(S[-1]+dS)

    ## 檢查是否有碰到上下界
        if (S[-1] <= y_barrier) & (not touch_N):
            touch_Y = True
            break
        if (S[-1] >= n_barrier) & (not touch_Y):
            touch_N = True
            break

    ## 檢查上下界的狀況已決定是否得到報酬
    if touch_Y & (not touch_N):
        get_payoff = True

    return get_payoff,S,V

get_payoff, S, V = monte_carlo(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sig
ma,n,T)


print('得到報酬:{}'.format(get_payoff))
x = np.arange(len(S))
plt.figure(figsize=(16,8))
plt.subplot(211)
plt.xlim((-5, xright))
plt.ylim((100, 115))
plt.plot(x,S)
plt.axhline(105, color= 'r')
plt.axhline(110, color= 'r')
plt.title('FX Spot',size=20)
plt.subplot(212)
plt.xlim((-5, xright))
plt.plot(x,V)
plt.title('Volatility',size=20)
```

```
得到報酬:True
```

Out[3]:

```
Text(0.5, 1.0, 'Volatility')
```



五、蒙地卡羅模擬及結果作圖(計算問題一)

In [4]:

```
result_list = []
S_list = []
V_list = []

for i in range(nos):
    result,S,V = monte_carlo(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma
,n,T)
    if result:
        result_list.append(1)
    else:
        result_list.append(0)
    S_list.append(S)
    V_list.append(V)
    if i % 10000 == 0:
        print("已經模擬{:^8d}次".format(i+1))

print('成功率 = {:>4f}%'.format(sum(result_list) / len(result_list)*100))
print("模擬結束")
```

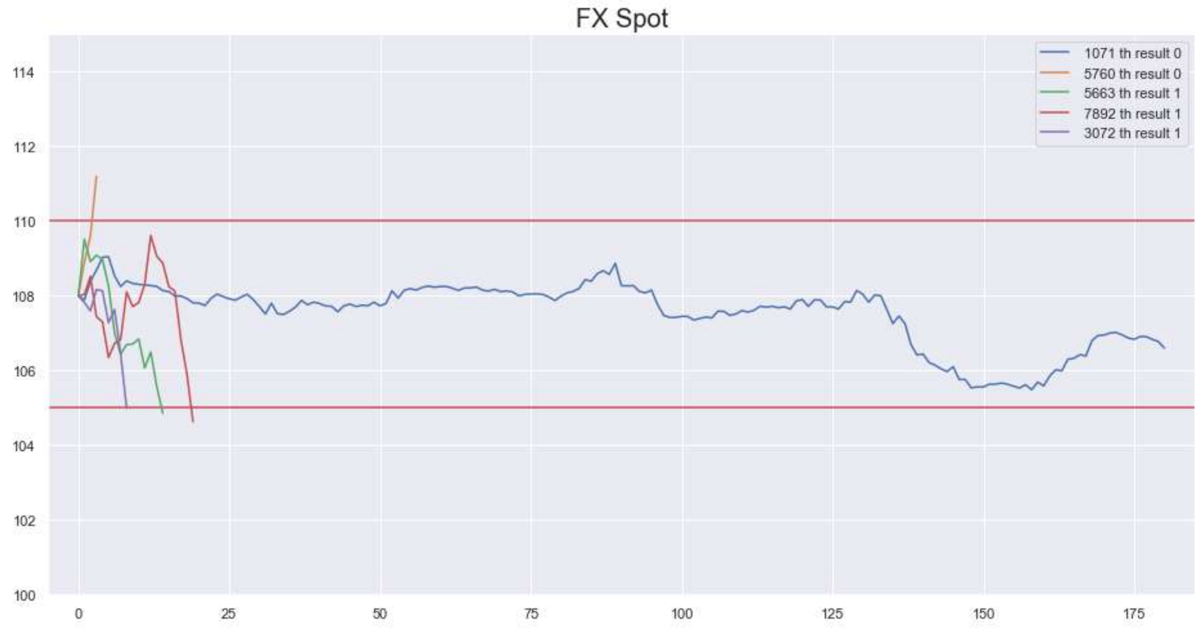
```
已經模擬   1    次
成功率 = 40.390000%
模擬結束
```

In [5]:

```
## 想做一個按鈕 可以選你要看幾次模擬結果
sample = np.random.randint(nos,size=5)
plt.figure(figsize=(16,8))
plt.axhline(105, color= 'r')
plt.axhline(110, color= 'r')
plt.title('FX Spot',size=20)
plt.xlim((-5, xright))
plt.ylim((100, 115))
for i in sample:
    x = np.arange(len(S_list[i]))
    plt.plot(x,S_list[i],label='{:^6d}th result {}'.format(i,result_list[i]))
plt.legend()
```

Out[5]:

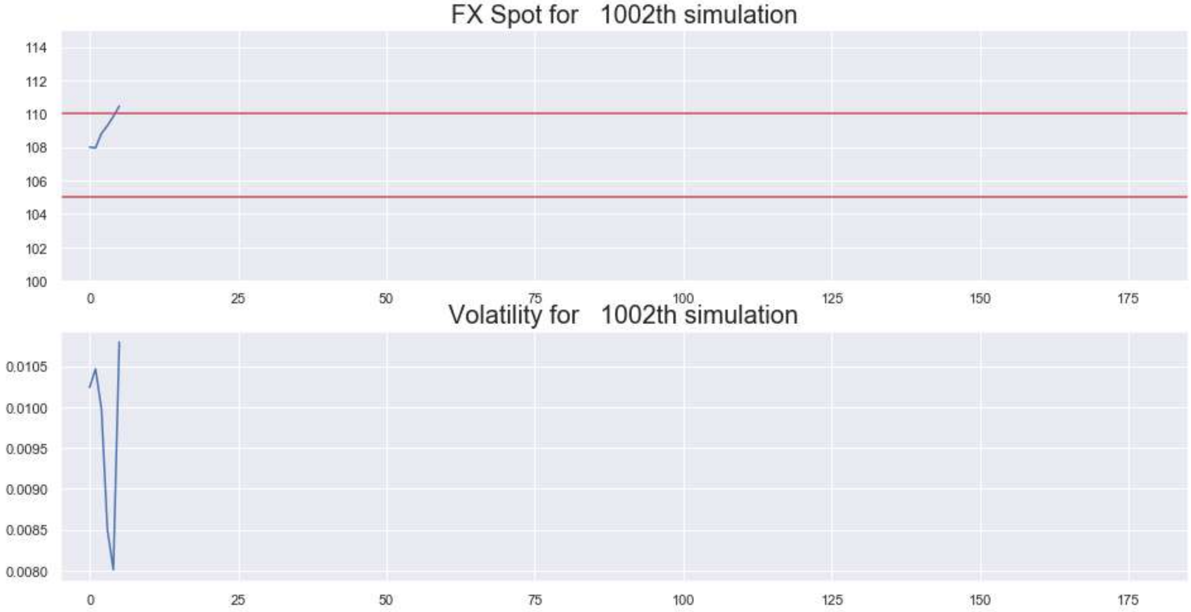
```
<matplotlib.legend.Legend at 0x1e02a17bb08>
```



In [6]:

```
## 想做一個按鈕  可以選你要看第幾次模擬結果
i = np.random.randint(nos,size=1)[0]
S = S_list[i]
V = V_list[i]
x = np.arange(len(S))
plt.figure(figsize=(16,8))
plt.subplot(211)
plt.xlim((-5, xright))
plt.ylim((100, 115))
plt.plot(x,S)
plt.axhline(105, color= 'r')
plt.axhline(110, color= 'r')
plt.title('FX Spot for {:>6d}th simulation'.format(i),size=20)
plt.subplot(212)
plt.xlim((-5, xright))
plt.plot(x,V)
plt.title('Volatility for {:>6d}th simulation'.format(i),size=20)
print('得到報酬:{}'.format(result_list[i]))
```

```
得到報酬:0
```



In [7]:

```
## 勝率及option價值
p = sum(result_list) / len(result_list)
(100*(1-p)+100*(1+0.03/2)*p)/(1+rf6m/2)
```

Out[7]:

```
100.42834290391734
```

六、Fair Value計算細節(計算問題一)

In [8]:

```
def op_value(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list):
    result_list = []
    S_list = []
    V_list = []

    for i in range(nos):
        seed = seed_list[i]
        result,S,V = monte_carlo(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,seed)
        if result:
            result_list.append(1)
        else:
            result_list.append(0)
        S_list.append(S)
        V_list.append(V)
    p = sum(result_list) / len(result_list)
    price = (100*(1-p)+100*(1+0.03/2)*p)/(1+rf6m/2)
    return price
```

In [9]:

```
seed_list = [None]*nos
op_value(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list)
```

Out[9]:

```
100.42804343323176
```

七、Delta計算細節(計算問題二)

試圖固定隨機種子以降低delta的波動，但成效不彰

In [59]:

```
def delta(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list):
    return (op_value(1.01*r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list)-op_value(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list))/0.01
```

In [83]:

```
delta_list = []
loop = 10
for i in range(loop):
    seed_list = np.random.randint(100000,size=nos)
    result = delta(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list)
    delta_list.append(result)
print("mean of delta : {}".format(np.array(delta_list).mean()))
print("std of delta : {}".format(np.array(delta_list).std()))
```

```
mean of delta : 0.15602422717918785
std of delta : 0.21878870017763752
```

八、Gamma計算細節(計算問題二)

參考 John Hull 10e P.479之公式

In [84]:

```
def gamma(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos):
    seed_list = np.random.randint(100000,size=nos)
    ans = (delta(1.01*r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list)-delta(0.99*r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list))/0.01
    return ans
```

In [85]:

```
gamma_list = []
loop = 10
for i in range(loop):
    result = gamma(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos)
    gamma_list.append(result)
print("mean of gamma : {}".format(np.array(gamma_list).mean()))
print("std of gamma : {}".format(np.array(gamma_list).std()))
```

```
mean of gamma : 6.438619739654428
std of gamma : 43.591453339958896
```

九、Vega計算細節

In [86]:

```
def vega(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list):
    ans = (op_value(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0+0.0001,kappa,theta,rho,sigma,n,T,nos,seed_list) - op_value(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,seed_list))/0.0001
    return ans
```

In [87]:

```
vega_list = []
loop = 10
for i in range(10):
    seed_list = np.random.randint(100000,size=100000)
    result = vega(r6m,fx_spot,y_barrier,n_barrier,rf6m,v0,kappa,theta,rho,sigma,n,T,nos,se
ed_list)
    vega_list.append(result)
print("mean of vega : {}".format(np.array(vega_list).mean()))
print("std of vega : {}".format(np.array(vega_list).std()))
```

```
mean of vega : -0.34439128842223
std of vega : 34.296850029470455
```

In [89]:

```
import pandas as pd
df = pd.DataFrame()
df['Delta'] = [np.array(delta_list).mean(),np.array(delta_list).std()]
df['Gamma'] = [np.array(gamma_list).mean(),np.array(gamma_list).std()]
df['Vega'] = [np.array(vega_list).mean(),np.array(vega_list).std()]
df.index = ['mean','std']
df.round(4)
```

Out[89]:

	Delta	Gamma	Vega
mean	0.1560	6.4386	-0.3444
std	0.2188	43.5915	34.2969

In [ ]: