Homework 3 Solution

March 14, 2021

1 0. Import Relevant Libraries

[1]:
```
from math import exp, sqrt, log
from scipy.stats import norm
import numpy as np
import pandas as pd
from scipy.optimize import root
import matplotlib.pyplot as plt
```

2 1. Caplet Pricing in Different Models

2.1 (a) Discount Rate

By the formula for discount rate, we know that

$$\begin{aligned} D(0, T+\delta) &= \exp\left\{-\int_0^{T+\delta} f(s)ds\right\}, \\ &= \exp\left\{-(T+\delta) \times f\right\}. \end{aligned}$$

Plugging $T = 1$, $\delta = 0.25$, $f = 1.25\%$ into the above equation, the following python code gives us the discount rate.

[2]:
```
T = 1
delta = 0.25
f = 0.0125

def DiscountFactor(S, T, f):
    D = exp(- (T - S) * f)
    return D

D = DiscountFactor(0, T + delta, f)

print("The discount rate is {:.10f}.".format(D))
```

```
The discount rate is 0.9844964370.
```

2.2 (b) Black-Scholes Model Price

The initial value of F_0 is

$$\begin{aligned} F_0 &= \frac{1}{\delta}\left(\exp\left\{\int_T^{T+\delta} f(0,s)ds\right\} - 1\right), \\ &= \frac{1}{\delta}\left(\exp\left\{\delta \times f\right\} - 1\right) \end{aligned}$$

Plugging $\delta = 0.25$, $f = 1.25\%$ into the above equation, the following python code gives us the F_0.

[3]:
```
F_0 = (1 / delta) * ( exp(delta * f) - 1)
print("F0 is {:.10f}.".format(F_0))
```

```
F0 is 0.0125195516.
```

By Black-Scholes formula for put option, we know that

$$\begin{aligned} E\left[(K - F_T)^+\right] &= N(-d_2) \times K - N(-d_1) \times F_0, \\ d_1 &= \frac{\log\left(\frac{F_0}{K}\right) + \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}, \\ d_2 &= \frac{\log\left(\frac{F_0}{K}\right) - \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}. \end{aligned}$$

Plugging into the caplet put option pricing formula,

$$P(K) = \delta D(0, T+\delta)\mathbb{E}\left[(K - F_T)^+\right],$$

we have the following python code to the price of the option on 1Y Libor by adapting the Black-Scholes formula.

[4]:
```
sigma = 0.15
K = 0.0125

def BlackScholesPut(F_0, K, T, sigma, delta, D):
    d_1 = (log(F_0 / K) + 0.5 * (sigma ** 2) * T) / (sigma * sqrt(T))
    d_2 = d_1 - sigma * sqrt(T)
    E = norm.cdf(- d_2) * K - norm.cdf(- d_1) * F_0
    P = delta * D * E

    return P

P_BS = BlackScholesPut(F_0, K, T, sigma, delta, D)
print("The price under Black Scholes Model is {:.10f}.".format(P_BS))
```

```
The price under Black Scholes Model is 0.0001816803.
```

2.3 (c) Bachelier or Normal Model setup

Since $\mathbb{E}[F_T] = F_0$, we can simply set $\sigma_n = \sigma \times F_0$, to approximate the instantaneous variance above. We have the following code to calculate σ_n.

[5]:
```
sigma_n = sigma * F_0
print("sigma_n is {:.10f}.".format(sigma_n))
```

```
sigma_n is 0.0018779327.
```

2.4 (d) Bachelier Model Price

[6]:
```
def BachelierPut(F_0, K, T, sigma, delta, D):
    d_minus = - (F_0 - K) / (sigma * sqrt(T))
    P = delta * D * sigma* sqrt(T)  * (d_minus * norm.cdf(d_minus) + norm.
 ↪pdf(d_minus))
    return P

P_norm = BachelierPut(F_0, K, T, sigma_n, delta, D)
print("The price under the Bachelier model is {:.10f}.".format(P_norm))
```

```
The price under the Bachelier model is 0.0001819969.
```

- The two prices are similar. This is because the incremental variance is almost the same in two SDEs.
- The difference between the two prices is shown belong, from which we can see that the price under the normal model is higher. This because that the Bachelier model allows negative rates, while Black-Scholes model only allows positive rates.

[7]:
```
difference = P_BS - P_norm
print("The difference betweem prices under diferent models is {:.10f}.".
 ↪format(difference))
```

```
The difference betweem prices under diferent models is -0.0000003166.
```

2.5 (e) Prices of a Series of Put Options with Different Strikes and the Associated Greeks

The delta of a put option is given by the following formula.

$$\Delta = \delta D(0, T+\delta)(N(d_1) - 1).$$

The gamma of a put option is given by the following formula.

$$\Gamma = \delta D(0, T+\delta)\frac{N'(d_1)}{F_0\sigma\sqrt{T}}.$$

The vega of a put option is given by the following formula.

$$V = \delta D(0, T+\delta)F_0\sqrt{T}N'(d_1).$$

The theta of a put option is given by the following formula.

$$\Theta = -\delta D(0, T+\delta)\frac{F_0\sigma N'(d_1)}{2\sqrt{T}}.$$

[8]:
```
Strikes = [0.005, 0.0075, 0.01, 0.0125]

Prices = []
Delta = []
Gamma = []
Vega = []
Theta = []

def DeltaPut(F_0, K, sigma, T, delta, D):
    d_1 = ( log( F_0 / K) + 0.5 * (sigma ** 2) * T ) / (sigma * sqrt(T))
    Delta_Put =  delta * D * ( norm.cdf(d_1) - 1 )

    return Delta_Put

def GammaPut(F_0, K, sigma, T, delta, D):
    d_1 = ( log( F_0 / K) + 0.5 * (sigma ** 2) * T ) / (sigma * sqrt(T))
    Gamma_Put = delta * D * ( norm.pdf(d_1) / (F_0 * sigma * sqrt(T)) )

    return Gamma_Put

def VegaPut(F_0, K, sigma, T, delta, D):
    d_1 = ( log( F_0 / K) + 0.5 * (sigma ** 2) * T ) / (sigma * sqrt(T))
    Vega_Put = delta * D * ( F_0 * sqrt(T) * norm.pdf(d_1) )

    return Vega_Put

def ThetaPut(F_0, K, sigma, T, delta, D):
    d_1 = ( log( F_0 / K) + 0.5 * (sigma ** 2) * T ) / (sigma * sqrt(T))
    Theta_Put = - delta * D * ( (F_0 * norm.pdf(d_1) * sigma) / (2 * sqrt(T)) )

    return Theta_Put

for K_0 in Strikes:
    Prices.append(BlackScholesPut(F_0, K_0, T, sigma, delta, D))
    Delta.append(DeltaPut(F_0, K_0, sigma, T, delta, D))
    Gamma.append(GammaPut(F_0, K_0, sigma, T, delta, D))
    Vega.append(VegaPut(F_0, K_0, sigma, T, delta, D))
    Theta.append(ThetaPut(F_0, K_0, sigma, T, delta, D))

data_Greeks_dict = {"Prices" : Prices, "Delta" : Delta, "Gamma" : Gamma, "Vega"␣
↪: Vega, "Theta" : Theta}
data_Greeks_df = pd.DataFrame(data_Greeks_dict)
print(data_Greeks_df)
```

```
         Prices         Delta         Gamma          Vega         Theta
0  2.134868e-14 -7.217245e-11  2.439647e-07  5.735823e-12 -4.301867e-13
1  2.906480e-08 -5.923641e-05  1.180617e-01  2.775734e-06 -2.081801e-07
2  1.213562e-05 -1.423937e-02  1.517264e+01  3.567220e-04 -2.675415e-05
```

```
3  1.816803e-04 -1.146850e-01  5.209545e+01  1.224810e-03 -9.186072e-05
```

We can see that the put option with strike 1.25% has the highest delta, gamma, and vega. The put option with strike 0.5% has the highest vega.

3 2. Stripping Caplet Volatilities

3.1 (a) Calculate the Price of Each Cap Using Black's Model

The prices for each cap is shown below.

[9]:
```
f = 0.01
delta = 0.25
F_0 = (1 / delta) * ( exp( delta * f) - 1 )


def Black_ATM(F_0, sigma, T):
    E = F_0 * (norm.cdf(0.5 * sigma * sqrt(T)) - norm.cdf(-0.5 * sigma *␣
 ↪sqrt(T)))

    return E

def ATM_Cap(start, Length, sigma):
    price = 0
    for i in range(int(Length / delta)):
        price = price + delta * DiscountFactor(0, start + (i + 1) * delta, f) *␣
 ↪Black_ATM(F_0, sigma, start + i * delta)

    return price

start = [1, 2, 3, 4, 5]
length = [2, 2, 2, 2, 2]
Black_Vol = [0.15, 0.2, 0.225, 0.225, 0.25]

data_cap_dict = {"Start" : start, "Length" : length, "Black Vol" : Black_Vol}
data_cap_df = pd.DataFrame(data_cap_dict)

prices = []


for i in range(5):
    start = data_cap_df.loc[i, "Start"]
    length = data_cap_df.loc[i, "Length"]
    sigma = data_cap_df.loc[i, "Black Vol"]

    prices.append(ATM_Cap(start, length, sigma))

data_cap_df["Price"] = prices
```

```
print(data_cap_df)
```

```
   Start  Length  Black Vol     Price
0      1       2      0.150  0.001582
1      2       2      0.200  0.002598
2      3       2      0.225  0.003357
3      4       2      0.225  0.003724
4      5       2      0.250  0.004478
```

3.2 (b) Extract the Black-Scholes ATM Implied Volatilities for Each Caplet

Here we use Bootstrapping to extract the Black-Scholes ATM Implied Volatilities. Suppose the implied volatility for caplets starting in the period 1Y - 3Y, 3Y - 4Y, 4Y - 5Y, 5Y - 6Y, 6Y - 7Y are σ_1, σ_2, σ_3, σ_4, σ_5.

By default, we have $\sigma_1 = 0.15$. To match the prices of different caps, we solve the σ_2, σ_3, σ_4, and σ_5 forwardly.

[10]:
```
sigma_1 = 0.15

def root_sigma(sigma_new, price, start, sigma_old):
    price_old = 0
    price_new = 0
    for i in range(4):
        price_old = price_old + delta * DiscountFactor(0, start - 1 + (i + 1) *␣
 ↪delta, f) * Black_ATM(F_0, sigma_old, start - 1 + i * delta)
        price_new = price_new + delta * DiscountFactor(0, start + (i + 1) *␣
 ↪delta, f) * Black_ATM(F_0, sigma_new, start + i * delta)

    return price_old + price_new - price

Implied_Vol = [sigma_1]
sigma = sigma_1
for i in range(3, 7):
    sol_sigma = root(root_sigma, sigma, args = (prices[i - 2], i, sigma))
    sigma = sol_sigma.x[0]
    Implied_Vol.append(sigma)

print(Implied_Vol)
```

```
[0.15, 0.2428191628142852, 0.20915576717102952, 0.23958757124653962,
0.25976769665246763]
```

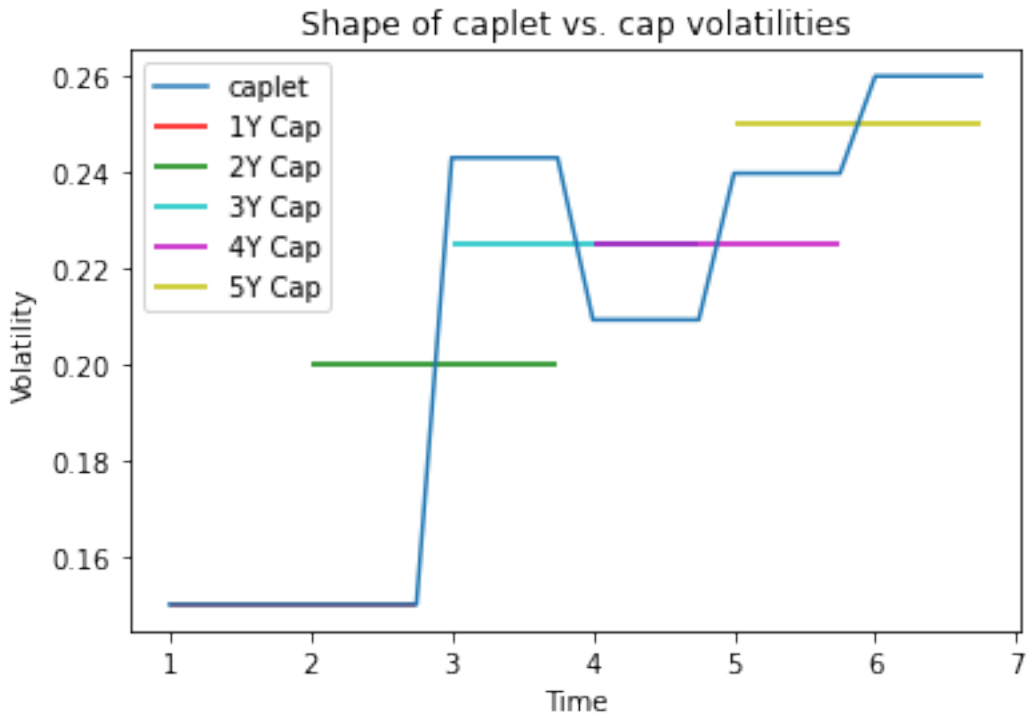
The shape of the caplet and cap volatilities is shown below.

[11]:
```
Time = np.linspace(1, 7, 24, endpoint = False)
IV = np.repeat(np.array(Implied_Vol), [8, 4, 4, 4, 4], axis = 0)

%matplotlib inline
```

```
plt.title("Shape of caplet vs. cap volatilities")
plt.xlabel("Time")
plt.ylabel("Volatility")
plt.plot(Time, IV, label = "caplet")
plt.hlines(Black_Vol[0], xmin = 1, xmax = 2.75, color = 'r', label = "1Y Cap")
plt.hlines(Black_Vol[1], xmin = 2, xmax = 3.75, color = 'g', label = "2Y Cap")
plt.hlines(Black_Vol[2], xmin = 3, xmax = 4.75, color = 'c', label = "3Y Cap")
plt.hlines(Black_Vol[3], xmin = 4, xmax = 5.75, color = 'm', label = "4Y Cap")
plt.hlines(Black_Vol[4], xmin = 5, xmax = 6.75, color = 'y', label = "5Y Cap")
plt.legend()
```

[11]: <matplotlib.legend.Legend at 0x20291812130>



From the above graph, we can see that cap implied volatilities can be viewed as a "weighted" average of each of the caplet immplied volatilities.