Prof. Dr. Natalie Packham
LV-Nr. 425093

Summer term 2019

Computational Finance and FinTech – Problem Set 7 with solutions

Exercise 1. Write functions that calculate the Black-Scholes price, delta, gamma and vega of a European call option.
Compute the price and sensitivies of a call option with parameters $S_0 = 100$, $K = 100$, $T = 1$, $\sigma = 20\%$ and $r = 5\%$. Give an interpretation of the sensitivities.

Solution 1.

```
import math
import numpy as np
import scipy as sp
import scipy.stats as scs
import matplotlib.pyplot as plt

def callprice(S,K,T,sigma,r):
    d1=(sp.log(S/K) + (r + 0.5 * sigma**2)*T) / (sigma * sp.sqrt(T))
    d2=(sp.log(S/K) + (r - 0.5 * sigma**2)*T) / (sigma * sp.sqrt(T))
    return S*scs.norm.cdf(d1) - math.exp(-r *T) * K * scs.norm.cdf(d2)

def delta(S, K, T, sigma, r):
    d1=(math.log(S/K) + (r + 0.5 * sigma**2)*T) / (sigma * math.sqrt(T))
    return scs.norm.cdf(d1)

def gamma(S, K, T, sigma, r):
    d1=(math.log(S/K) + (r + 0.5 * sigma**2)*T) / (sigma * math.sqrt(T))
    return scs.norm.pdf(d1) / (S * sigma * math.sqrt(T))

def vega(S, K, T, sigma, r):
    d1=(math.log(S/K) + (r + 0.5 * sigma**2)*T) / (sigma * math.sqrt(T))
    return S * math.sqrt(T) * scs.norm.pdf(d1)
```

The output for the specified call option is:

Price	10.4506
Delta	0.6368
Gamma	0.0188
Vega	37.5240

Interpretation:

- Delta: Trading 0.6368 units of the stock in the opposite direction of the call option exposure eliminates the risk from stock price changes (for a small time step). If the stock prices changes by 1 cent, then the option price changes by approximately 0.6368 cents. Let's assume a long position in the call option. If the stock price goes up by 1 EUR, then the call prices changes, then the call price changes as follows:

  ```
  callprice(101, 100, 1, 0.2, 0.05) - callprice(100, 100, 1, 0.2, 0.05)
  ```

  which gives 0.6461. Likewise if the stock price goes down by 1 EUR:

```
callprice(99, 100, 1, 0.2, 0.05) - callprice(100, 100, 1, 0.2, 0.05)
```

gives -0.6274. Holding -0.6368 units of the stock gives an overall P&L of 0.0093, resp. 0.0094 if the stock price goes up or down. (A zero P&L is achieved if the hedge portfolio is continuously rebalanced.)

- Gamma is the sensitivity of Delta, i.e., if the stock prices changes by 1 cent, then Delta changes by approximately 0.0188 cents.
- Vega is the sensitivity of σ ("model risk"), i.e., if the volatility changes by 1 percentage point (e.g. from 20% to 21%), then the option price changes by approximately $37.5240 \cdot 0.01 = 0.37524$ cents. Indeed,

```
callprice(100, 100, 1, 0.21, 0.05) - callprice(100, 100, 1, 0.2, 0.05)
```

gives 0.3757, and

```
callprice(100, 100, 1, 0.19, 0.05) - callprice(100, 100, 1, 0.2, 0.05)
```

gives -0.3747.

Exercise 2. Consider the setup from the previous exercise: a stock with $\sigma = 20\%$ trades at $S_0 = 100$, the risk-free interest rate is $r = 5\%$. Consider a long position in a call option with strike $K = 100$ and $T = 1$.

(a) What position in the underlying stock is required to make the position delta-neutral? Verify by consider a ± 1 EUR stock price changes.

(b) Now assume that a call option with $K = 110$, $T = 1$ is liquidly traded in the market and therefore can be used for hedging. What position in the traded option and in the underlying make the position both delta-neutral and gamma-neutral? Again, verify by considering ± 1 EUR stock price change scenarios.

(c) A second liquidly traded option is available; the parameters are $K = 90$ and $T = 1$. How can the position be made delta-, gamma- and vega-neutral?

Hint: The following linear approximation of a position value change ΔV can be applied

$$\Delta V \approx \frac{\partial V}{\partial S}\Delta S + \frac{\partial V}{\partial \sigma}\Delta\sigma + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(\Delta S)^2,$$

so making the portfolio gamma-, resp. vega-neutral, requires simply to match the respective sensitivities.

Solution 2.

(a) See Exercise 1.

(b) Let C denote the long call option, S the stock price and H_1 the call option used for hedging. To make the position gamma- and delta-neutral requires that

$$\left(\frac{\partial C}{\partial S} + x\frac{\partial S}{\partial S} + y\frac{\partial H_1}{\partial S}\right)\Delta S + \frac{1}{2}\left(\frac{\partial^2 C}{\partial S^2} + y\frac{\partial^2 H_1}{\partial S^2}\right)(\Delta S)^2 = 0,$$

(Note that the stock price has a gamma of zero.) This can be expressed as a system of linear equations that needs to be solved for x and y:

$$\begin{aligned}\frac{\partial C}{\partial S} + x\frac{\partial S}{\partial S} + y\frac{\partial H_1}{\partial S} &= 0\\ \frac{\partial^2 C}{\partial S^2} + x\underbrace{\frac{\partial^2 S}{\partial S^2}}_{=0} + y\frac{\partial^2 H_1}{\partial S^2} &= 0\end{aligned}$$

To enter this into `np.linalg.solve`, re-write this as

$$\begin{aligned}x\frac{\partial S}{\partial S} + y\frac{\partial H_1}{\partial S} &= -\frac{\partial C}{\partial S}\\ x\underbrace{\frac{\partial^2 S}{\partial S^2}}_{=0} + y\frac{\partial^2 H_1}{\partial S^2} &= -\frac{\partial^2 C}{\partial S^2}\end{aligned}$$

Using the numerical solver from Problem Set 6

```
np.linalg.solve(np.array([[1, delta(100, 110, 1, 0.2, 0.05)], [0, gamma(100
                                        , 110, 1, 0.2, 0.05)]]),\
                np.array([-delta(100, 100, 1, 0.2, 0.05),-gamma(100, 100, 1
                                                    , 0.02, 0.05)]))
```

gives

```
array([-0.2105, -0.9482])
```

If the stock price appreciates by 1 EUR, the hedged position changes by -0.0001 and it changes by 0.0001 if the stock price depreciates by 1 EUR. Code:

```
(callprice(101 ,100, 1, 0.2, 0.05) - 0.2105 * 101 - 0.9482 * callprice(101,
                                        110, 1, 0.2, 0.05)) \
- (callprice(100 ,100, 1, 0.2, 0.05) - 0.2105 *100 - 0.9482 * callprice(100
                                        , 110, 1, 0.2, 0.05))
```

```
(callprice(99 ,100, 1, 0.2, 0.05) - 0.2105 * 99 - 0.9482 * callprice(99,
                                        110, 1, 0.2, 0.05)) \
- (callprice(100 ,100, 1, 0.2, 0.05) - 0.2105 *100 - 0.9482 * callprice(100
                                        , 110, 1, 0.2, 0.05))
```

(c) Making the position gamma-, delta- and vega-neutral requires solving for x, y, z:

$$\begin{aligned}x\frac{\partial S}{\partial S} + y\frac{\partial H_1}{\partial S} + z\frac{\partial H_2}{\partial S} &= -\frac{\partial C}{\partial S}\\ x\underbrace{\frac{\partial^2 S}{\partial S^2}}_{=0} + y\frac{\partial^2 H_1}{\partial S^2} + z\frac{\partial^2 H_2}{\partial S^2} &= -\frac{\partial^2 C}{\partial S^2}\\ x\underbrace{\frac{\partial S}{\partial \sigma}}_{=0} + y\frac{\partial H_1}{\partial \sigma} + z\frac{\partial H_2}{\partial \sigma} &= -\frac{C}{\partial \sigma}\end{aligned}$$

Python code:

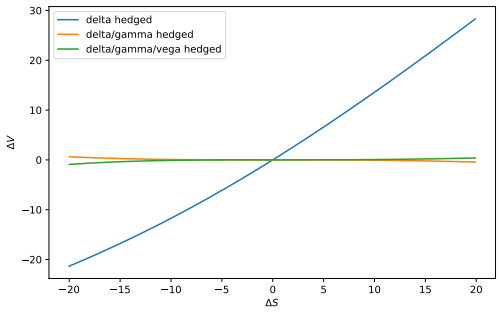
```
np.linalg.solve(np.array([[1, delta(100, 110, 1, 0.2, 0.05), delta(100, 90,
                                        1, 0.2, 0.05)], \
                          [0, gamma(100, 110, 1, 0.2, 0.05), gamma(100, 90,
                                                                   1, 0.2,
                                                                  0.05)], \
                          [0, vega(110, 110, 1, 0.2, 0.05), vega(100, 90, 1
                                                                 , 0.2, 0.
                                                                 05)]]),\
                 np.array([-delta(100, 100, 1, 0.2, 0.05),-gamma(100, 100, 1
                                                          , 0.02, 0.05),\
                           -vega(100, 100, 1, 0.2, 0.05)]))
```

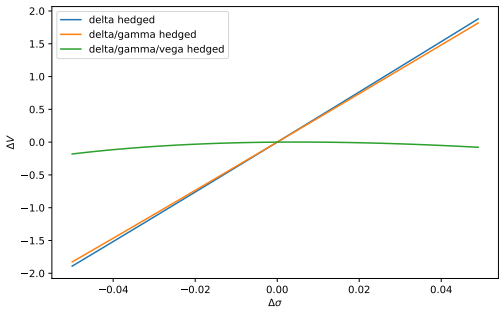
Output:

```
array([ 0.4817, -0.    , -1.3815])
```

Value changes for ± 1 EUR change in the stock price: 0.00005, resp. -0.00005. Value change for $\pm$ 1 percentage point in volatility: -0.00489 and -0.00527.

The following two plots show the changes in the hedged positions.





Exercise 3. Create a plot of the call price surface as a function of time t and stock price S_t. Parameters you could use: $K = 100$, $T = 1$, $\sigma = 20\%$ and $r = 5\%$.

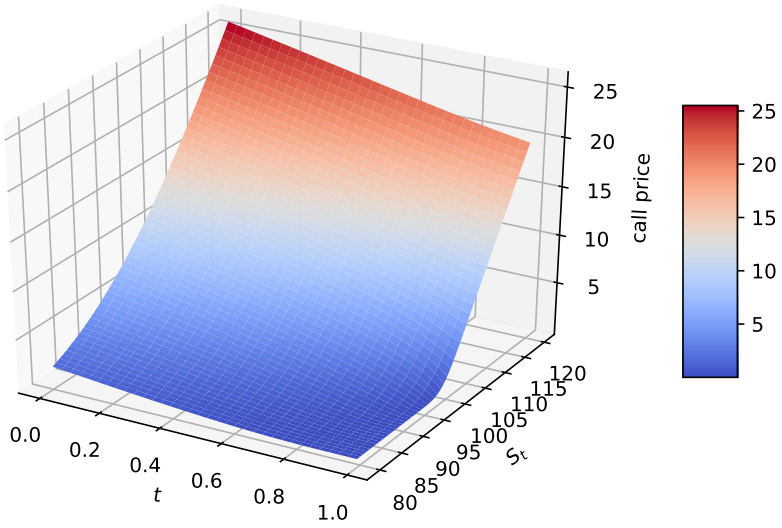
Solution 3.

```
t = np.arange(0,1,0.01)
s = np.arange(80, 120, 0.25)

c = np.empty((len(s), len(t)))
for i in range(len(s)):
    for j in range(len(t)):
        c[i,j] = callprice(s[i], 100, 1-t[j], 0.2, 0.05)

t, s = np.meshgrid(t, s)

from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(8, 5))
ax = fig.gca(projection='3d')    # set up canvas for 3D plotting
surf = ax.plot_surface(t, s, c, rstride=3, cstride=3,\
                       cmap=plt.cm.coolwarm, linewidth=0.5,\
                       antialiased=True)
ax.set_ylabel('$S_t$')
ax.set_xlabel('$t$')
ax.set_zlabel('call price')
fig.colorbar(surf, shrink=0.5, aspect=5);
plt.savefig('call.pdf')
```



Exercise 4. Use Monte Carlo simulation to price a **compound option**. A compound option is an option on an option, e.g. a call option to acquire a call option. The option to be priced is a call on a call with the following parameters: $S_0 = 100$, $\sigma = 20\%$, $r = 5\%$, $T_1 = 1$, $T_2 = 2$, $K_1 = 10$, $K_2 = 100$. Set the simulation interval to $n = 100,000$ and calculate a 95%-confidence interval.

(Hint: The analytical price of the option is 8.8465.

Solution 4.

```
n=100000
w = np.random.standard_normal(n)
s = 100 * np.exp((0.05 - 0.5 * 0.2**2) * 1 + 0.2 * sp.sqrt(1) * w)
p = (callprice(s, 100, 1, 0.2, 0.05)-10)
y = np.exp(-0.05 *1) * (p + abs(p))/2

y_m = []
y_cfl = []
y_cfu = []
for i in range(1000, n+1, 1000):
    y_m.append(np.mean(y[:i]))
    y_cfl.append(np.mean(y[:i] - 1.96 * np.std(y[:i])/np.sqrt(i)))
    y_cfu.append(np.mean(y[:i] + 1.96 * np.std(y[:i])/np.sqrt(i)))

plt.plot(range(1000,n+1,1000), y_m, 'b', \
         range(1000,n+1,1000), y_cfl, 'g', range(1000,n+1,1000), y_cfu, 'g');
plt.xlabel('number of simulations');
plt.ylabel('estimates');

# Confidence interval and point estimate
[y_cfl[-1], y_m[-1], y_cfu[-1]]
```

Output: `[8.806034039276415, 8.887505475259168, 8.968976911241917]`

Exercise 5. Load the data from `tr_eikon_eod_data.csv` (see Chapter 5 on Financial Time Series and Problem Set 3). Create a data frame with log-returns. Now run the following two trading strategies, each with an initial investment of 100 EUR:

- *Buy-and-hold*: Invest half of the investment amount in each Apple and Amazon and hold it.
- *Constant mix*: Invest half of the investment amount in ach Apple ana Amazon and rebalance the portfolio each day to maintain weights of 50%.

Plot the portfolio values in a time-series graph.

Solution 5.

```
import numpy as np
import pandas as pd
from pylab import mpl, plt
plt.style.use('seaborn')

filename = './data/tr_eikon_eod_data.csv'
data = pd.read_csv(filename, index_col=0, parse_dates=True)
data.dropna(inplace=True)
rets = (data-data.shift(1)) / data.shift(1)

rets.dropna(inplace=True)

buyhold = 50 * (1+rets['AAPL.O']).cumprod() + 50 * (1+rets['GS.N']).cumprod()

constmix = pd.Series(index=rets.index)

constmix.iloc[0]=100;
value=100;
for t in rets.index:
    constmix.loc[t] = value * (1+0.5 * rets['AAPL.O'].loc[t] + 0.5 * rets['GS.N
                                                    '].loc[t])
    value=constmix.loc[t]

plt.figure()
plt.plot(buyhold)
plt.plot(constmix);
```