5QQMN534: Algorithmic Finance

Week4: Data Visualisation

Appendix B: Option Example

Class Definition

- The following presents a class definition for a European call option in the Black-Scholes-Merton (1973) model. The class-based implementation is an alternative to the one based on functions as presented in "Python Script":

```python
#
# Valuation of European call options in Black-Scholes-Merton model
# incl. vega function and implied volatility estimation
# -- class-based implementation
#
# Python for Finance, 2nd ed.
# (c) Dr. Yves J. Hilpisch
#
from math import log, sqrt, exp
from scipy import stats


class bsm_call_option(object):
    ''' Class for European call options in BSM model.

    Attributes
    ==========
    S0: float
        initial stock/index level
    K: float
        strike price
    T: float
        maturity (in year fractions)
    r: float
        constant risk-free short rate
    sigma: float
        volatility factor in diffusion term

    Methods
    =======
    value: float
        returns the present value of call option
    vega: float
        returns the vega of call option
    imp_vol: float
        returns the implied volatility given option quote
    '''

    def __init__(self, S0, K, T, r, sigma):
        self.S0 = float(S0)
        self.K = K
        self.T = T
        self.r = r
        self.sigma = sigma

    def value(self):
        ''' Returns option value.
        '''
        d1 = ((log(self.S0 / self.K) +
               (self.r + 0.5 * self.sigma ** 2) * self.T) /
              (self.sigma * sqrt(self.T)))
        d2 = ((log(self.S0 / self.K) +
               (self.r - 0.5 * self.sigma ** 2) * self.T) /
              (self.sigma * sqrt(self.T)))
        value = (self.S0 * stats.norm.cdf(d1, 0.0, 1.0) -
                 self.K * exp(-self.r * self.T) * stats.norm.cdf(d2, 0.0, 1.0))
        return value

    def vega(self):
        ''' Returns vega of option.
        '''
        d1 = ((log(self.S0 / self.K) +
               (self.r + 0.5 * self.sigma ** 2) * self.T) /
              (self.sigma * sqrt(self.T)))
        vega = self.S0 * stats.norm.pdf(d1, 0.0, 1.0) * sqrt(self.T)
        return vega

    def imp_vol(self, C0, sigma_est=0.2, it=100):
        ''' Returns implied volatility given option price.
        '''
        option = bsm_call_option(self.S0, self.K, self.T, self.r, sigma_est)
        for i in range(it):
            option.sigma -= (option.value() - C0) / option.vega()
        return option.sigma
```

Class Usage

- This class can be as follows:

```
In [1]: from bsm_option_class import *

In [2]: o = bsm_call_option(100., 105., 1.0, 0.05, 0.2)
        type(o)
Out[2]: bsm_option_class.bsm_call_option

In [3]: value = o.value()
        value
Out[3]: 8.021352235143176

In [4]: o.vega()
Out[4]: 39.67052380842653

In [5]: o.imp_vol(C0=value)
Out[5]: 0.2
```

Visualisation1

- The option class can also be used to visualize, for example, the value and vega of the option for different strikes and maturities. It is, in the end, one of the major advantages of having an analytical option pricing formula available.

- The following Python code generates the option statistics for different maturity-strike combinations:


```
In [6]: import numpy as np
        maturities = np.linspace(0.05, 2.0, 20)
        strikes = np.linspace(80, 120, 20)
        T, K = np.meshgrid(strikes, maturities)
        C = np.zeros_like(K)
        V = np.zeros_like(C)
        for t in enumerate(maturities):
            for k in enumerate(strikes):
                o.T = t[1]
                o.K = k[1]
                C[t[0], k[0]] = o.value()
                V[t[0], k[0]] = o.vega()
```

Visualisation2

- First, a look at the option values. Figure B-1 (next slide) presents the value surface for the European call option:


```
In [7]: from pylab import cm, mpl, plt
        from mpl_toolkits.mplot3d import Axes3D
        mpl.rcParams['font.family'] = 'serif'
        %matplotlib inline

In [8]: fig = plt.figure(figsize=(12, 7))

                ax = fig.gca(projection='3d')
                surf = ax.plot_surface(T, K, C, rstride=1, cstride=1,
                            cmap=cm.coolwarm, linewidth=0.5, antialiased=True)
                ax.set_xlabel('strike')
                ax.set_ylabel('maturity')
                ax.set_zlabel('European call option value')
                fig.colorbar(surf, shrink=0.5, aspect=5);
```

Visualisation3

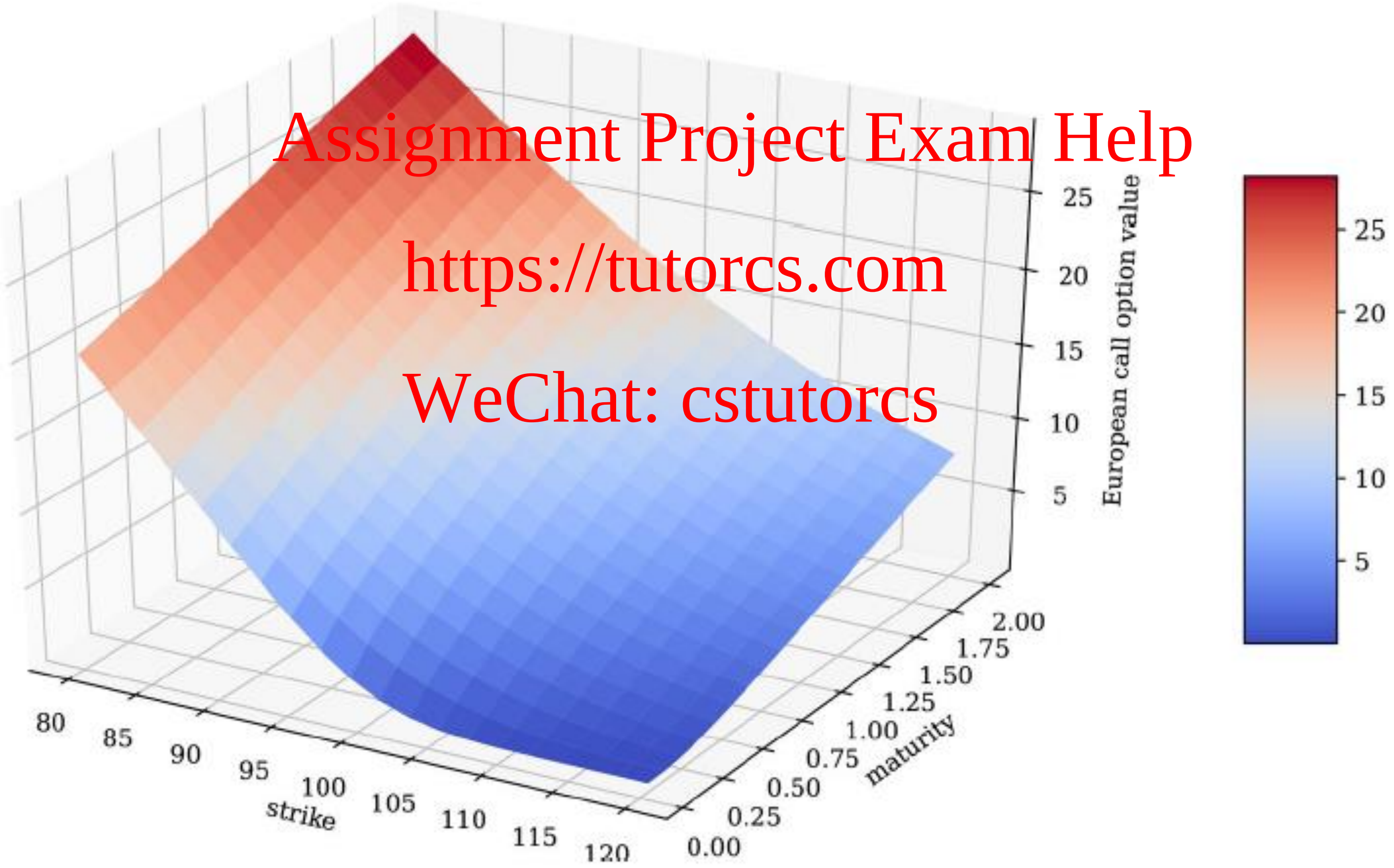

Figure B-1. Value surface for European call option

Visualisation4

```
In [9]: fig = plt.figure(figsize=(12, 7))
        ax = fig.gca(projection='3d')
        surf = ax.plot_surface(T, K, V, rstride=1, cstride=1,
                    cmap=cm.coolwarm, linewidth=0.5, antialiased=True)
        ax.set_xlabel('strike')
        ax.set_ylabel('maturity')
        ax.set_zlabel('Vega of European call option')
        fig.colorbar(surf, shrink=0.5, aspect=5);
```

- Second, a look at the vega values. Figure B-2 presents the vega surface for the European call option:

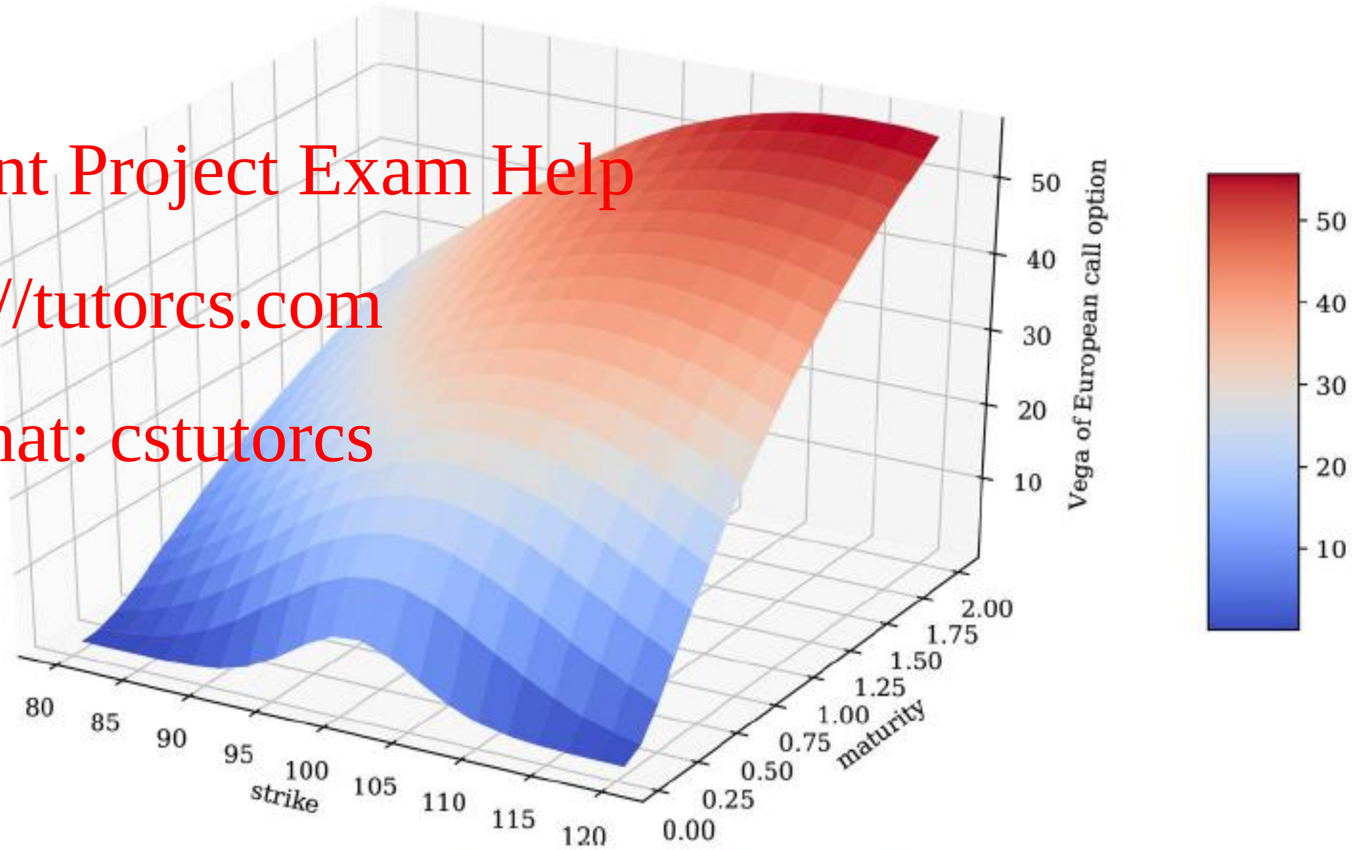

Figure B-2. Vega surface for European call option