Basic & Advanced Options Greeks Calculator

The notebook consists of the following:

- Understanding The Black-Scholes Option Pricing Model.
- Understaning The Basic & Advanced Option Greeks.
- Option Greeks Calculator.

First, lets import all the relevant Python packages required.

[1]:
```python
import numpy as np
import math
from scipy.stats import norm
```

Understanding The Black-Scholes Option Pricing Model.

The Black-Scholes Option pricing model states several assumptions through which the differential equation is derived.Firstly, the markets follow a random walk and therefore are unpredictable.Second, the markets have no arbitrage opportunities.Third, there is zero presence of market frictions like taxes and transaction costs.Fourth, the returns of the underlying assset follow a normal distribution and prices of assets follow a geometric Brownian motion along with drift and some noise.Fifth, the volatility of the underlying asset is known at time 0.Sixth, the model can only be used for European options where the right to exercise the option is only at maturity.Lastly, it also assumes that the underlying stock does not pay an dividends during the life of the option.The Black-Scholes model takes several parameter in to consideration for the pricing of the options which are: Current Stock Price, Strike Price of the option, Maturity period, Risk-free interest rate and volatility.The above assumptions are the reasosn why the Black_Scholes model can not price options accurately in reality since these assumptions are voided in actual stock markets.

Now we know that the Black-Scholes European Call Option Pricing Formula is given by:

$$C_0 = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

Where

- C_0 is the European call option price at time-0.
- S_0 is the current price of the underlying, in our case lets take a stock.
- K is the exercise price of the option.
- r is the risk-free interest rate.
- T is the time to maturity
- Φ is the cumulative distribution function of the Normal distribution.
- $d_1 = \frac{\frac{logS_0}{K} + (r+0.5\sigma^2)T}{\sigma\sqrt{T}}$
- $d_2 = d_1 - \sigma\sqrt{T}$
- σ is the volatility of the underlying stock.

Now the Black-Scholes European Put Option Formula can be derived using the Put-Call Parity $C_0 + Xe^{-rT} = P_0 + S_0$ is given by:

$$P_0 = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1)$$

Where

- P_0 is the European put option price at time-0.
- S_0 is the current price of the underlying, in our case lets take a stock.
- K is the exercise price of the option.
- r is the risk-free interest rate.
- T is the time to maturity
- Φ is the cumulative distribution function of the Normal distribution.
- $d_1 = \frac{\frac{logS_0}{K} + (r+0.5\sigma^2)T}{\sigma\sqrt{T}}$
- $d_2 = d_1 - \sigma\sqrt{T}$
- σ is the volatility of the underlying stock.

Understanding Basic Option Greeks

Option greeks are risk parameters that provide useful information for the fluctuations in the option prices due to several factors like volatility, time, interest rates and dividend yields.There are two types of greeks, basic and advanced, where the advanced greeks are just a mix and different order greeks based of the basic greeks.Lets first look at the basic greeks and what exactly they entail.Delta shows the rate of change in the option price with respect to the change in the undelying asset.Gamma is the second order derivative of delta which represents the rate of change in delta with respect to the change in the underlying asset.Vega is the rate of change in the option price with respect to the change in volatility of the underlying asset.Theta is the rate of change in the option with respect to time.Lastly, Rho depicts the rate of change in the option price with respect to change in the risk-free interest rates.The most important greeks are Delta, Vega and Theta as the majority proportion of the option price is dependent on the underlying asset prices, volatility and time.An option with a short maturity period will suffer from theta decay a lot due to its nature as the time premium component of the option price completely melts to 0 at expiry.However, for a short period option contract the greek Rho has less significance as the probability of changes in the risk-free interest rates in the short run in very low and thus the changes have a less impact in the short run.Although, the Rho will increase as the maturity period of the option increases as it is now more succeptible to interest rate changes.

Basic Option Greeks For European Call Option & Their Formulations:

Delta: $\Delta = \frac{\partial C_0}{\partial S_0} = \phi(d_1)$

Gamma: $\Gamma = \frac{\partial^2 C_0}{\partial {S_0}^2} = \frac{\phi(d_1)}{S_0\sigma\sqrt{T}}$

Vega: $\mathcal{V} = \frac{\partial C_0}{\partial \sigma} = S_0\phi(d_1)\sqrt{T}$

Theta: $\Theta = \frac{\partial C_0}{\partial t} = -\frac{S_0\phi d_1\sigma}{2\sqrt{T}} - rKe^{-rT}\Phi(d_2)$

Rho: $\rho = \frac{partial C_0}{\partial r} = KTe^{-rT}\Phi(d_2)$

Where:

- ϕ is the Probability Density Function of the standard normal distribution.

Basic Option Greeks For European Put Option & Their Formulations:

Delta: $\Delta = \frac{\partial P_0}{\partial S_0} = \phi(d_1) - 1$

Gamma: $\Gamma = \frac{\partial^2 P_0}{\partial {S_0}^2} = \frac{\phi(d_1)}{S_0\sigma\sqrt{T}}$

Vega: $\mathcal{V} = \frac{\partial P_0}{\partial \sigma} = S_0\phi(d_1)\sqrt{T}$

Theta: $\Theta = \frac{\partial P_0}{\partial t} = -\frac{S_0\phi d_1\sigma}{2\sqrt{T}} - rKe^{-rT}\Phi(-d_2)$

Rho: $\rho = \frac{\partial C_0}{\partial r} = -KTe^{-rT}\Phi(-d_2)$

Where:

- ϕ is the Probability Density Function of the standard normal distribution

Now lets create a class consisting the functions for the greeks for faster computations.

[17]:
```python
class European_Call_Basic_Greeks:

    def bs_d1(self, S0, K, T, sigma, r):
        d1 = (np.log(S0/K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        return d1

    def bs_d2(self, S0, K, T, sigma, r):
        d2 = self.bs_d1(S0, K, T, sigma, r) - sigma * np.sqrt(T)
        return d2

    def delta(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return norm.cdf(d1)

    def gamma(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return norm.pdf(d1) / (S0 * sigma * np.sqrt(T))

    def vega(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return S0 * norm.pdf(d1) * np.sqrt(T)

    def theta(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return - ((S0 * norm.pdf(d1) * sigma) / (2 * np.sqrt(T))) - r * K * np.exp(- r * T) * norm.cdf(d2)

    def rho(self, S0, K, T, sigma, r):
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return K * T * np.exp(- r * T) * norm.cdf(d2)
```

Now using the above class and functions lets compute an exemplary option's greeks.

[59]:
```python
S0 = 100
K = 100
T = 2
sigma = 0.25
r = 0.05
```

[58]:
```python
print(f'Delta = {European_Call_Basic_Greeks().delta(S0, K, T, sigma, r)}')
print(f'Gamma = {European_Call_Basic_Greeks().gamma(S0, K, T, sigma, r)}')
print(f'Vega = {European_Call_Basic_Greeks().vega(S0, K, T, sigma, r)}')
print(f'Theta = {European_Call_Basic_Greeks().theta(S0, K, T, sigma, r)}')
print(f'Rho = {European_Call_Basic_Greeks().rho(S0, K, T, sigma, r)}')
```

```
Delta = 0.6771052869398673
Gamma = 0.010152726911428257
Vega = 50.76363455714129
Theta = -5.625899806889205
Rho = 98.126905882715
```

Now lets do the same for European Put Option.

[57]:
```python
class European_Put_Basic_Greeks:

    def bs_d1(self, S0, K, T, sigma, r):
        d1 = (np.log(S0/K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        return d1

    def bs_d2(self, S0, K, T, sigma, r):
        d2 = self.bs_d1(S0, K, T, sigma, r) - sigma * np.sqrt(T)
        return d2

    def delta(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return norm.cdf(d1) - 1

    def gamma(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return norm.pdf(d1) / (S0 * sigma * np.sqrt(T))

    def vega(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return S0 * norm.pdf(d1) * np.sqrt(T)

    def theta(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return - ((S0 * norm.pdf(d1) * sigma) / (2 * np.sqrt(T))) + r * K * np.exp(- r * T) * norm.cdf(-d2)

    def rho(self, S0, K, T, sigma, r):
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return - K * T * np.exp(- r * T) * norm.cdf(-d2)
```

[56]:
```python
print(f'Delta = {European_Put_Basic_Greeks().delta(S0, K, T, sigma, r)}')
print(f'Gamma = {European_Put_Basic_Greeks().gamma(S0, K, T, sigma, r)}')
print(f'Vega = {European_Put_Basic_Greeks().vega(S0, K, T, sigma, r)}')
print(f'Theta = {European_Put_Basic_Greeks().theta(S0, K, T, sigma, r)}')
print(f'Rho = {European_Put_Basic_Greeks().rho(S0, K, T, sigma, r)}')
```

```
Delta = -0.32289471306013273
Gamma = 0.010152726911428257
Vega = 50.76363455714129
Theta = -1.1017127167094074
Rho = -82.84057772447692
```

Understanding Advanced Option Greeks

As mentioned before the advanced greeks are based of the basic greeks and are now a more popular risk metric due to the advancements in computational finance where professionals are now able to explore more complex dimensions of the option pricing.Lets look at what they consist of.Lambda depicts the percentage change in the option value per percentage change in the underlying price.Epsilon demonstrates the percentage change in the option price per percentage change in the dividend yield of the underlying stock.Vanna is the change in vega with respect to percentage change in underlying stock price.Charm is a option greek which considers two parameters at once which is the rate of change in delta over time and thus its formulation given in the code cell below.Vomma is the second order derivative of Vega i.e. the change in Vega with respect to change in the volatility.Veta illustrates the rate of change in Vega with respect to time.Speed is the third order derivative of the Delta i.e. change in Gamma with respect to a change in the underlying stock price.Zomma measures the rate of change in Gamma with respect to the change in volatility and is thus a third order greek.Color is very similar to Zomma except for the fact that it measures the rate of change in Gamma with respect to change in time.Moreover, Ultima is another third order greek of volatility which demonstrates the sensitivity of Vomma with respect to change in volatilty.As we can observe above that advanced greeks are mainly constructed for factors like the movement in the underlying stock and volatility while considering time with it.Traders often employ these create intraday trading strategies and to construct longterm option portfolio to benefit from the hedging characteristic.

Advanced Option Greeks For European Call Option & Their Formulations:

Lambda: $\lambda = \frac{\partial C_0}{\partial S_0} \times \frac{S_0}{C_0} = \Delta\frac{S_0}{C_0}$

Epsilon: $\epsilon = \frac{\partial C_0}{\partial q} = -STe^{-qT}\Phi(d_1)$

Where:

- q is the divident yield of the underlying stock.

Advanced Option Greeks For European Put Option & Their Formulations:

Lambda: $\lambda = \frac{\partial C_0}{\partial P_0} \times \frac{S_0}{P_0} = \Delta\frac{S_0}{P_0}$

Epsilon: $\epsilon = \frac{\partial P_0}{\partial q} = STe^{-qT}\Phi(-d_1)$

Where:

- q is the divident yield of the underlying stock.

Advanced Option Greeks For Both European Call & Put Options & Their Formulations:

Vanna: $\frac{\partial \mathcal{C}}{\partial S_0} = -\phi(d_1)\frac{d_2}{\sigma}$

Charm: $\frac{\partial^2 C_0}{\partial T\partial S_0} = -\phi(d_1)\frac{2rT - d_2\sigma\sqrt{(T)}}{2T\sigma\sqrt{(T)}}$

Vomma: $\frac{\partial^2 C_0}{\partial \sigma^2} = \mathcal{V}\frac{d_1 d_2}{\sigma}$

Veta: $\frac{\partial^2 C_0}{\partial \sigma\partial T} = -S_0\phi(d_1)\sqrt{T}(\frac{rd_1}{\sigma\sqrt{(T)}} - \frac{1+d_1d_2}{2T})$

Speed: $\frac{\partial^3 C_0}{\partial S_0^3} = \frac{\partial \Gamma}{\partial S_0} = \frac{\phi(d_1)}{S_0^2\sigma\sqrt{T}}(\frac{d_1}{\sigma\sqrt{T}} + 1)$

Zomma: $\frac{\partial^3 C_0}{\partial S_0^2\partial\sigma} = \frac{\partial\Gamma}{\partial\sigma} = \frac{\phi(d_1)(d_1d_2-1)}{S_0\sigma^2\sqrt{T}}$

Color: $\frac{\partial^3 C_0}{\partial S_0^2\partial T} = \frac{\partial\Gamma}{\partial T} = \frac{\phi(d_1)}{2S_0T\sigma\sqrt{T}}[2T + 1 + \frac{2rT - d_2\sigma\sqrt{T}}{\sigma\sqrt{T}}d_1]$

Ultima: $\frac{\partial^3 C_0}{\partial\sigma^3} = \frac{\partial vomma}{\partial\sigma} = \frac{\mathcal{V}}{\sigma^2}[d_1d_2(1 - d_1d_2) + d_1^2 + d_2^2]$

[41]:
```python
class European_Call_Advanced_Greeks:

    def bs_d1(self, S0, K, T, sigma, r):
        d1 = (np.log(S0/K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        return d1

    def bs_d2(self, S0, K, T, sigma, r):
        d2 = self.bs_d1(S0, K, T, sigma, r) - sigma * np.sqrt(T)
        return d2

    def _lambda(self, S0, K, T, sigma, r):
        delta = European_Call_Basic_Greeks().delta(S0, K, T, sigma, r)
        return delta * (S0 / C)

    def epsilon(self, S0, K, T, sigma, r, q):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return - S0 * T * np.exp(- q * T) * norm.cdf(d1)
```

[42]:
```python
class European_Put_Advanced_Greeks:

    def bs_d1(self, S0, K, T, sigma, r):
        d1 = (np.log(S0/K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        return d1

    def bs_d2(self, S0, K, T, sigma, r):
        d2 = self.bs_d1(S0, K, T, sigma, r) - sigma * np.sqrt(T)
        return d2

    def _lambda(self, S0, K, T, sigma, r):
        delta = European_Put_Basic_Greeks().delta(S0, K, T, sigma, r)
        return delta * (S0 / C)

    def epsilon(self, S0, K, T, sigma, r, q):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        return S0 * T * np.exp(- q * T) * norm.cdf(-d1)
```

[53]:
```python
class European_Advanced_Greeks_For_Both:

    def bs_d1(self, S0, K, T, sigma, r):
        d1 = (np.log(S0/K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        return d1

    def bs_d2(self, S0, K, T, sigma, r):
        d2 = self.bs_d1(S0, K, T, sigma, r) - sigma * np.sqrt(T)
        return d2

    def vanna(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return norm.pdf(d1) * d2 / sigma

    def charm(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return - norm.pdf(d1) * ((2 * r * T - d2 * sigma * np.sqrt(T)) / (2 * T * sigma * np.sqrt(T)))

    def vomma(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return S0 * norm.pdf(d1) * np.sqrt(T) * ((d1 * d2)/sigma)

    def veta(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return - S0 * norm.pdf(d1) * np.sqrt(T) * (r * d1 / (sigma * np.sqrt(T)) - (1 + d1 * d2) / (2 * T))

    def zomma(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return norm.pdf(d1) * (d1 * d2 - 1) / (S0 * sigma**2 * np.sqrt(T))

    def color(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        return norm.pdf(d1) / (2 * S0 * T * sigma * np.sqrt(T)) * (2 * T + 1 + ((2 * r * T - d2 * sigma * np.sqrt(T)) / sigma * np.sqrt(T)) * d1)

    def ultima(self, S0, K, T, sigma, r):
        d1 = self.bs_d1(S0, K, T, sigma, r)
        d2 = self.bs_d2(S0, K, T, sigma, r)
        vega = European_Call_Basic_Greeks.vega(self, S0, K, T, sigma, r)
        return - vega / sigma**2 * (d1 * d2 * (1 - d1 * d2) + d1**2 + d2**2)
```

Lets use the example below to compute its advanced greeks that we saw and created above.

[54]:
```python
S0 = 100
K = 100
T = 2
sigma = 0.25
r = 0.05
q = 0.05
```

[45]:
```python
print(f'European Call Epsilon = {European_Call_Advanced_Greeks().epsilon(S0, K, T, sigma, r, q)}')
print(f'European Put Epsilon = {European_Put_Advanced_Greeks().epsilon(S0, K, T, sigma, r, q)}')
```

```
European Call Epsilon = -122.5340399146334
European Put Epsilon = 58.4334436925585
```

[55]:
```python
print(f'Vanna = {European_Advanced_Greeks_For_Both().vanna(S0, K, T, sigma, r)}')
print(f'Charm = {European_Advanced_Greeks_For_Both().charm(S0, K, T, sigma, r)}')
print(f'Vomma = {European_Advanced_Greeks_For_Both().vomma(S0, K, T, sigma, r)}')
print(f'Veta = {European_Advanced_Greeks_For_Both().veta(S0, K, T, sigma, r)}')
print(f'Zomma = {European_Advanced_Greeks_For_Both().zomma(S0, K, T, sigma, r)}')
print(f'Color = {European_Advanced_Greeks_For_Both().color(S0, K, T, sigma, r)}')
print(f'Ultima = {European_Advanced_Greeks_For_Both().ultima(S0, K, T, sigma, r)}')
```

```
Vanna = 0.15229090367142378
Charm = -0.041245453077677297
Vomma = 9.898908738642547
Veta = 10.009954189236298
Zomma = -0.03863112589798452
Color = 0.013763290419304932
Ultima = -218.38388677395787
```

This concludes the notebook.