Introduction

Fischer Black, Myron Scholes and Robert Merton made significant advances in the development of options pricers with their papers published in 1973.

The development of a transparent and reasonably robust options pricing model underpinned the transformational growth of the options market over the decades to follow.

In this document the key assumptions of the Black Scholes model are defined, the analytical solutions to the Black Scholes differential equations stated.

Having shown the solutions, this document shows the development of the VBA user defined functions required for implementation of Black Scholes analytical pricing solutions using Excel.

The key benefits of defining a custom function in Excel for using Black Scholes in VBA include

- Saving time – custom functions enable reusability of the function across different worksheets when placed in an add-in
- Reduced risk of user error
- Functions are also extensible

The Black Scholes Model of Stock Prices

According to the Black Scholes model, the price path of stocks is defined by the following stochastic partial differential equation

\[ dS = (r - q - \frac{1}{2} \sigma^2) dt + \sigma dz \]

where \( dz \) is a standard Brownian motion, defined by \( dz = \epsilon \cdot \sqrt{dt} \) where \( \epsilon \) is a standard normal random variable; \( dS \) is the change in stock price, \( r \) is the risk-free interest rate, \( q \) is the dividend of the stock, \( \sigma \) the volatility of the stock.

The model implies that \( dS/S \) follows a normal distribution with mean \( r - q - \frac{1}{2} \sigma^2 \), and standard deviation \( \sigma \epsilon \sqrt{dt} \).

As such the price at time \( 0 < t \leq T \) is given by

\[ S_T = S_0 \cdot \exp \left( (r - q - \frac{1}{2} \sigma^2) t + \sigma \epsilon \sqrt{t} \right) \]

The price of a European call option is given by \( \max(St - K, 0) \) where \( St \) is the final stock price at expiry.
Black Scholes Option Pricing Assumptions

The Black Scholes Merton differential equation is derived using the following assumptions:

1. The stock price follows the model above
2. Short selling with full use of proceeds is allowed
3. No transactions costs or taxes, all securities perfectly divisible
4. No riskless arbitrage opportunities exist
5. Security trading is continuous
6. The risk-free rate of interest, $r$, is constant and the same for all maturities
7. There are no dividends during the life of the asset*

*Black Scholes results can be extended to cover European call and put options on dividend-paying stocks. This is done by using the Black-Scholes formula with the stock price reduced by the present value of the dividends during the life of the option, and the volatility equal to the volatility of the stock price net of the present value of the dividends.

Black Scholes Stochastic Differential Equations

Given the above assumptions, Black and Scholes use Ito’s lemma to derive a differential equation for the price of an option, then create a riskless portfolio of one short call option and long delta amount of shares, which under the no-arbitrage assumptions, must earn the risk-free rate of return. Using this model they arrive at the following differential equation:

$$\frac{df}{dt} + rS\frac{df}{S} + \frac{1}{2} \sigma^2 S^2 \frac{d^2f}{S^2} = rf$$

Using either risk-neutral valuation or solving the partial differential equation using the boundary conditions that $f = \max(S-K, 0)$ at expiry for a call, and $f=\max(K-S,0)$ for a put, the following formulas are derived. These are the solutions to the Black Scholes equation.

$$C = S_0 N(d_1) - K \exp(-rT) N(d_2)$$

$$P = K \exp(-rT) N(-d_2) - S_0 N(-d_1)$$

$$D_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma \sqrt{T}}$$

$$D_2 = \frac{\ln(S_0/K) + (r - \sigma^2/2)T}{\sigma \sqrt{T}} = d_1 - \sigma \sqrt{T}$$
How to build a Black Scholes VBA Option Pricer

The **Black Scholes VBA** Custom Function

The function takes in 6 parameters, S, K, r, q, t, sigma, N (number of simulations) and should return the call price and the put price. To return the two values, we need to create a function that returns an array.

The main issue with reusability of functions is performance. Custom functions are slower than Excel’s built-in functions, but their performance can be optimised.

Excel VBA has late and early binding depending on the data type declarations. Late binding slows down VBA code execution noticeably over many different simulation cycles. As such, early binding is recommended. This is done by not defining the variables of the function as variant, and by explicitly defining each and every variable’s data type, including the array returned by the function.

So where a ‘lazy’ function declaration may look like

Function Black_Scholes (S, K, r, q, t, sigma, N)

The proper way to define the function would be

Function Black_Scholes (S as double, K as double, r as double, q as double, t as double, sigma as double, N as long) as double

As such, the full code would be as follows:

Function Black_Scholes(ByVal S As Double, ByVal K As Double, ByVal r As Double, ByVal q As Double, ByVal t As Double, ByVal sigma As Double) As Double

Dim d1 As Double
Dim d2 As Double
Dim a As Double
Dim b_call As Double
Dim b_put As Double
Dim c As Double
Dim call_price As Double
Dim put_price As Double

a = Log(S / K)

b_call = (r - q + 0.5 * sigma ^ 2) * t
The **Black Scholes VBA** Custom Function (cont)

\[
\begin{align*}
b_{\text{put}} &= (r - q - 0.5 \cdot \sigma^2) \cdot t \\
c &= \sigma \cdot \sqrt{t} \\
d_1 &= \frac{(a + b_{\text{call}})}{c} \\
d_2 &= \frac{(a + b_{\text{put}})}{c} \\
call_price &= S \cdot \text{WorksheetFunction.NormSDist}(d_1) - K \cdot \exp(-r \cdot t) \cdot \text{WorksheetFunction.NormSDist}(d_2) \\
put_price &= K \cdot \exp(-r \cdot t) \cdot \text{WorksheetFunction.NormSDist}(-d_2) - S \cdot \text{WorksheetFunction.NormSDist}(-d_1)
\end{align*}
\]

`Black_Scholes = Array(call_price, put_price)`

End Function

---

**Advanced VBA – Get Excel to add user tips**

In order to enhance the usability of the function one could add a subroutine to add user tips and show it in the list of Excel functions.

Sub UDF_Function_Descriptions()

Dim sFn_Description As String

sFn_Description = "Returns the price of a vanilla call and a put option in two cells, according to the Black Scholes model." & _

vbLf "Select the output cells, enter the formula, press Ctrl + Shift + Enter." & _

vbLf "Black Scholes(Asset Price, Strike, Time Interval (years), Risk free interest rate (%), Asset yield (%), Volatility (%))"

Application.MacroOptions Macro:="Black_Scholes", Description:=sFn_Description, Category:=1

End Sub

---

**How VBA Developer.net Can Save You Time and money**

You can get complete Excel apps from [VBA Developer.net](https://www.vba-developer.net) containing the code in this document, customisation, VBA development of any Excel, Access and Outlook apps, as well as C# and C++ add-ins and technical documentation.

[Visit VBA Developer.net](https://www.vba-developer.net)

**Examples of VBA documents from VBA Developer.net**

How to build a Black Scholes VBA Option Pricer

How to build a Black Scholes C# Option Pricer
How to build a Black Scholes VBA Option Pricer

How to build a Black Scholes VBA Option Pricer for FX Options

How to build a Black Scholes VBA Option Pricer for Equity Options

How to build a Black Scholes VBA Option Pricer using Monte Carlo Simulation

How to build a Black Scholes VBA Option Pricer for Binary Options

How to build a Black Scholes VBA Option Pricer for Equity Barrier Options

How to build a Black Scholes VBA Option Pricer for Exotic Asian Options

How to build a Black Scholes VBA Option Pricer for Exotic Lookback Options

How to build an Equity Option Pricer using the Binomial Tree in Excel VBA

How to code a Cholesky Decomposition in VBA (Numerical Methods for Excel)

3 ways to sort in VBA

How to Code a Multivariate Value at Risk (VaR) VBA Monte Carlo Simulation

How To Code the Newton-Raphson Method in Excel VBA

How to Model Volatility Smiles, Volatility Term Structure and the Volatility Surface in Excel VBA

How To Write Use Cases for a Portfolio Reporting VBA Tool

How To Write a User Interface Model For a Portfolio Reporting VBA Tool

How To Create a Semantic Object Model For a Portfolio Reporting VBA Tool

How To Normalise a Database For VBA Apps

How To Create a Database using SQL Scripts for a Portfolio Reporting VBA App

How to Write Stored Procedures in SQL/Access/VBA for a Portfolio Reporting VBA App

How to Use Cursors in SQL for a Portfolio Reporting VBA Tool

How to Move Data from Access to Excel with SQL for a Portfolio Reporting VBA App
How to build a Black Scholes VBA Option Pricer

Portfolio Reporting VBA Tool: Inserting Data into SQL/Access Databases from Excel

Portfolio Reporting VBA Tool: Connecting Excel with SQL & Access Databases

How To Design Classes For an Option Pricer in VBA: UML Concepts

How To Design Classes for Object Orientated VBA Programming