

Spreadsheet Option Functions Available with *Funadamentals of Derivatives Markets*

Robert L. McDonald

February 18, 2008

Contents

1	Introduction	2
1.1	Operating Systems	2
1.2	Programming Language	2
1.3	Spreadsheets	2
1.4	Using the Add-ins	3
1.5	A Note on Array Functions	3
2	Black-Scholes Functions	4
2.1	Prices	4
2.2	Greeks	4
2.3	Black-Scholes Array Functions	5
2.4	Black Formula	5
2.5	Perpetual American Options	6
3	Binomial Functions	6
4	Exotic Options	7
4.1	“Vanilla” Barrier Options	7
4.2	Other Barrier Options	8
4.2.1	Cash-or-Nothing	8
4.2.2	Asset-or-Nothing	10
4.3	Asian Options	11

1 Introduction

This is documentation for the option-pricing functions available in the spreadsheets accompanying *Fundamentals of Derivatives Markets*. There are versions of the spreadsheet that run with Microsoft Office and with OpenOffice (www.openoffice.org). The functioning of the spreadsheets should be identical under the two systems. However, the Excel version of the spreadsheet has had years of use. The OpenOffice version is new and there may be bugs or glitches; please let me know if you encounter any problems.

If you are not familiar with OpenOffice, you may find it worthwhile to become acquainted with it. OpenOffice is a rapidly maturing free office suite that runs on Windows, Linux, and Apple OS X.

1.1 Operating Systems

The functions available with this book will run under Windows (using Microsoft Office or OpenOffice), OS X (using Microsoft Office 2004 or earlier, or OpenOffice), and Linux (Openoffice).

As of 2008, Microsoft no longer supports Visual Basic for Applications under Microsoft Office on OS X. If you use an Intel-based Mac, the only way to run these macros is to use Office 2004 or to obtain OpenOffice for the Macintosh.

1.2 Programming Language

All of the functions described here are user-defined functions for which the code is accessible and user-modifiable. There are two versions of the spreadsheets:

- *Excel*: These spreadsheets are written in VBA and can be edited using the Visual Basic editor
- *OpenOffice Calc*: These spreadsheets are written in StarBasic and can be edited using the built-in basic editor in OpenOffice.

You will find that there are minimal differences between VBA and StarBasic *except* in the routines that directly access spreadsheet functions (such as reading from and writing to cells, and using built-in functions).

1.3 Spreadsheets

The following spreadsheets come with the book:

OptAll2Basic.xls This is an Excel spreadsheet that provides examples of most (not all) of the pricing functions described here.

OptAll2Basic.xla This is an add-in version of the Excel spreadsheet.

OptAll2Basic.ods This is an OpenOffice version of the similarly-named Excel spreadsheet.

Data.xls This spreadsheet provides sample stock price and option price data.

1.4 Using the Add-ins

Spreadsheets defined as add-ins have the extension “.xla”. Add-ins are typically used to provide additional functionality for Excel. The add-ins provided on this CD-Rom allow you to have the option pricing functions automatically available for use when you run Excel.

To use the add-ins, you first need to place this file where Excel can find it. By default, according to the Excel help file:

Add-ins are stored by default in one of the following places:

- The Library folder or one of its subfolders in the Microsoft Office/Office folder.
- The Documents and Settings/<user name>/Application Data/Microsoft/AddIns folder.

You can place the add-in elsewhere, but you will then have to browse for it to use it.

To install the add-in, select Tools|Add-ins and check the box for “OptAll2.xla”. If the add-in you want does not appear in the list, you need to select “browse” and locate it yourself. Once you install the add-in, you will have access to all of the functions in this spreadsheet, without needing to open the spreadsheet explicitly.

You can create your own add-in easily from any spreadsheet by selecting File|Save As and then specifying “Excel add-in (*.xla)”.

Please note that when you use add-in functions in a spreadsheet, the functions are not saved with your spreadsheet and do not move from computer to computer with the spreadsheet. If you want to have a “portable” spreadsheet you should start with one of the option pricing spreadsheets, modify it as you please, and then save it under a different name.

1.5 A Note on Array Functions

Some of the functions described here are array functions. This means that the output of the function can be in more than one cell. In order to enter an array function, do the following:

1. highlight the output range
2. press the F2 key to edit the first cell in the range
3. enter the formula
4. press the control, shift, and enter keys simultaneously.

Once you have completed these steps, you will notice that cells in the array range exhibit the formula you entered surrounded by curly braces. Once an array function is entered in a range, that range can only be edited or deleted as a single entity. Excel and OpenOffice will prohibit you from editing one cell of an array.

2 Black-Scholes Functions

The following symbol definitions are used in this section: “s” = stock price, “k” = strike price, “v” = volatility (annualized), “r” = interest rate (continuously-compounded, annualized), “t” = time to expiration (years), and “d” = dividend yield (continuously-compounded, annualized).

2.1 Prices

Function Definition	Function Description
BSCall(s, k, v, r, t, d)	<i>European call option price</i>
BSPut(s, k, v, r, t, d)	<i>European put option price</i>

2.2 Greeks

BSCallDelta(s, k, v, r, t, d)	<i>European call delta</i>
BSPutDelta(s, k, v, r, t, d)	<i>European put delta</i>
BSCallGamma(s, k, v, r, t, d)	<i>European call gamma</i>
BSPutGamma(s, k, v, r, t, d)	<i>European put gamma</i>
BSCallVega(s, k, v, r, t, d)	<i>European call vega</i>
BSPutVega(s, k, v, r, t, d)	<i>European put vega</i>
BSCallRho(s, k, v, r, t, d)	<i>European call rho</i>
BSPutRho(s, k, v, r, t, d)	<i>European put rho</i>
BSCallTheta(s, k, v, r, t, d)	<i>European call theta</i>
BSPutTheta(s, k, v, r, t, d)	<i>European put theta</i>
BSCallPsi(s, k, v, r, t, d)	<i>European call psi</i>
BSPutPsi(s, k, v, r, t, d)	<i>European put psi</i>
BSCallElast(s, k, v, r, t, d)	<i>European call elasticity</i>
BSPutElast(s, k, v, r, t, d)	<i>European put elasticity</i>

<code>BSCallImpVol(s, k, v, r, t, d, c)</code>	<i>Implied volatility for European call</i> The option price is entered as “c”; the volatility entered does not matter.
<code>BSPutImpVol(s, k, v, r, t, d, c)</code>	<i>Implied volatility for European put</i> The option price is entered as “c”; the volatility entered does not matter.
<code>BSCallImpS(s, k, v, r, t, d, c)</code>	<i>Implied stock price for a given European call option price</i>
<code>BSPutImpS(s, k, v, r, t, d, c)</code>	<i>Implied stock price for a given European put option price</i>

2.3 Black-Scholes Array Functions

The functions in this section are all array functions (up to 16×16) and have an extra string, “x” as a parameter. This parameter controls the output, with a “c” denoting “call” and “p” denoting “put”, and “d”, “g”, “r”, “v”, “t”, and “e” denoting delta, gamma, rho, vega, theta, and elasticity. A “+” means continue placing the output in the same row and a “/” means move to the next row. For example, the string “cp+cd+cg/pp+pd+pg” will output a 2 row, 3 column array containing a call price, delta, and gamma in the first row, with the same information for a put in the second row.

<code>BS(s,k,v,r,t,d,x)</code>	<i>Black-Scholes prices and Greeks</i>
<code>BS_Text(x)</code>	<i>Array function providing text description of the items in the string “x”</i>
<code>BS_Text_Greek(x)</code>	<i>Array function providing text description of the Greeks corresponding to items in the string “x”</i>

2.4 Black Formula

The Black formula gives the price of an option where a futures contract is the underlying asset. The Black formula is the Black-Scholes formula with the dividend yield set equal to the interest rate. It is implemented as an array function, exactly like the Black-Scholes array functions described in Section 2.3. In particular, “x” is a format string, controlling the output of the function.

<code>BlackFormula(s,k,v,r,t,d,x)</code>	<i>Black-Scholes prices and Greeks</i>
<code>Black_Text(x)</code>	<i>Array function providing text description of the items in the string “x”</i>

Black_Text_Greek(x)

Array function providing text description of the Greeks corresponding to items in the string "x"

2.5 Perpetual American Options

There is no simple pricing formula for American options except when the options are infinitely-lived. These functions are array functions, returning the option price and the stock price at which the option should optimally be exercised, in that order. The results may be returned in either a horizontal or vertical array.

Function CallPerpetual(s, k, v, r, d) *Perpetual American call*

Function PutPerpetual(s, k, v, r, d) *Perpetual American put*

3 Binomial Functions

The following symbol definitions are used in this section: "s" = stock price, "k" = strike price, "v" = volatility (annualized), "r" = interest rate (continuously-compounded, annualized), "t" = time to expiration (years), "d" = dividend yield (continuously-compounded, annualized), "N" = number of binomial steps, "opstyle" = option style (0 = European, 1 = American), and "vest" = the period of time during which the option cannot be exercised (after this time, exercise is permitted). If "N", the number of binomial steps, is set less than or equal to 0, then N is internally reset to be 100.

By default, all binomial calculations are done using the forward tree used in Chapter 10. There is a constant in the VBA code called "TreeType". If set equal to 0 (the default), all calculations are done with a forward tree. If equal to 1, a CRR tree is used, and if equal to 2, a Jarrow-Rudd tree is used.

The functions in this section are all array functions, returning the option price, along with delta, gamma, and theta, in that order.

So, for example, if you just enter the BinomCall function in a single cell, it will return the option price. If you enter it as an array function spanning two cells, you will get the price and delta, etc... The array can be entered either horizontally or vertically.

BinomCall(s, k, v, r, t, d, opstyle, N) *Binomial call*

BinomPut(s, k, v, r, t, d, opstyle, N) *Binomial put*

BinomCallBermudan(s, k, v, r, t, d, opstyle, N, vest) *Binomial Bermudan call*

BinomPutBermudan(s, k, v, r, t, d, opstyle, N, vest) *Binomial Bermudan put*

<pre>BinomOptFixedDivCall(s, k, v, r, t, div_amt, div_h, div_next, opstyle, N)</pre>	<p><i>Binomial call price when underlying asset pays dollar dividends in amount <code>div_amt</code>, with the first payment coming at time <code>div_next</code>, with dividends spaced <code>div_h</code> apart.</i></p> <p><i>The function</i></p>
<pre>BinomOptFixedDivPut(s, k, v, r, t, div_amt, div_h, div_next, opstyle, N)</pre>	<p><i>Binomial put price when underlying asset pays dollar dividends in amount <code>div_amt</code>, with the first payment coming at time <code>div_next</code>, with dividends spaced <code>div_h</code> apart.</i></p>

4 Exotic Options

The following symbol definitions are used in this section: “s” = stock price, “k” = strike price, “v” = volatility (annualized), “r” = interest rate (continuously-compounded, annualized), “t” = time to expiration (years), “d” = dividend yield (continuously-compounded, annualized), “rho” = correlation coefficient, and “H” = barrier.

4.1 “Vanilla” Barrier Options

For ordinary barrier puts and calls, the (descriptive) names are of the form

Option Type + Barrier Type

For example, the pricing function for an up-and-in call is “CallUpIn”.

<code>CallDownIn(s, k, v, r, t, d, h)</code>	<i>Down-and-in call</i>
<code>CallDownOut(s, k, v, r, t, d, h)</code>	<i>Down-and-out call</i>
<code>CallUpIn(s, k, v, r, t, d, h)</code>	<i>Up-and-in call</i>
<code>CallUpOut(s, k, v, r, t, d, h)</code>	<i>Up-and-out call</i>
<code>PutDownIn(s, k, v, r, t, d, h)</code>	<i>Down-and-in put</i>
<code>PutDownOut(s, k, v, r, t, d, h)</code>	<i>Down-and-out put</i>
<code>PutUpIn(s, k, v, r, t, d, h)</code>	<i>Up-and-in put</i>
<code>PutUpOut(s, k, v, r, t, d, h)</code>	<i>Up-and-out put</i>

4.2 Other Barrier Options

For most other barrier options, the function names are mnemonic, and have three parts:

Payoff Type + Barrier Type + Option Type

In particular,

- The payoff type is one of
 - Cash** the payoff is \$1
 - Asset** the payoff is one unit of the asset
- The barrier type is one of
 - DI** “down-and-in”, in other words the stock price is initially above the barrier, which must be hit in order for the payoff to be received.
 - DO** “down-and-out”, in other words the stock price is initially above the barrier, which must not be hit in order for the payoff to be received.
 - UI** “up-and-in”, in other words the stock price is initially below the barrier, which must be hit in order for the payoff to be received.
 - UO** “up-and-out”, in other words the stock price is initially below the barrier, which must not be hit in order for the payoff to be received.
- The option type is one of
 - Call** in order for the payoff to be received, the asset price must be above the strike at expiration
 - Put** in order for the payoff to be received, the asset price must be below the strike at expiration

For example, an option that pays \$1 if the stock price exceeds the strike price at expiration, provided that the barrier, below the initial stock price, has not been hit would have the name

CashDOCall = $\underbrace{\text{Cash}}_{\text{Payoff of \$1}}$ + $\underbrace{\text{DO}}_{\text{if the barrier, below the initial stock price, has not been hit}}$ + $\underbrace{\text{Call}}_{\text{and the stock price exceeds the strike at expiration}}$

4.2.1 Cash-or-Nothing

<p>CashCall(s, k, v, r, t, d)</p> <p>CashPut(s, k, v, r, t, d)</p>	<p><i>Cash-or-nothing call</i> Receive \$1 if $s_t > k$ at expiration</p> <p><i>Cash-or-nothing put</i> Receive \$1 if $s_t < k$ at expiration</p>
----------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CashDICall(s, k, v, r, t, d, h)	<i>Cash-or-nothing down-and-in call</i> Receive \$1 if $s_t > k$ at expiration and the barrier $H < s$ has been hit.
CashDOCall(s, k, v, r, t, d, h)	<i>Cash-or-nothing down-and-out call</i> Receive \$1 if $s_t > k$ at expiration and the barrier $H < s$ has not been hit.
CashDOPut(s, k, v, r, t, d, h)	<i>Cash-or-nothing down-and-out put</i> Receive \$1 if $s_t < k$ at expiration and the barrier $H < s$ has not been hit.
CashDIPut(s, k, v, r, t, d, h)	<i>Cash-or-nothing down-and-out put</i> Receive \$1 if $s_t < k$ at expiration and the barrier $H < s$ has been hit.
CashUICall(s, k, v, r, t, d, h)	<i>Cash-or-nothing up-and-in call</i> Receive \$1 if $s_t > k$ at expiration and the barrier $H > s$ has been hit.
CashUOCall(s, k, v, r, t, d, h)	<i>Cash-or-nothing up-and-out call</i> Receive \$1 if $s_t > k$ at expiration and the barrier $H > s$ has not been hit.
CashUOPut(s, k, v, r, t, d, h)	<i>Cash-or-nothing up-and-out put</i> Receive \$1 if $s_t < k$ at expiration and the barrier $H > s$ has not been hit.
CashUIPut(s, k, v, r, t, d, h)	<i>Cash-or-nothing up-and-in put</i> Receive \$1 if $s_t < k$ at expiration and the barrier $H > s$ has been hit.
DR(s, v, r, t, d, h)	<i>Down rebate</i> Receive \$1 at the time the barrier, $H < s$, is hit.
UR(s, v, r, t, d, h)	<i>Up rebate</i> Receive \$1 at the time the barrier, $H > s$, is hit.
DRDeferred(s, v, r, t, d, h)	<i>Deferred down rebate</i> Receive \$1 at expiration if prior to expiration the barrier, $H < s$, is hit.
URDeferred(s, v, r, t, d, h)	<i>Deferred up rebate</i> Receive \$1 at expiration if prior to expiration the barrier, $H < s$, is hit.

4.2.2 Asset-or-Nothing

Note that there is no difference between an option that pays H if the share price hits H , and an option that pays a share if the share price hits H .

<code>AssetCall(s, k, v, r, t, d)</code>	<i>Asset-or-nothing call</i> Receive one unit of the asset if $s_t > k$ at expiration
<code>AssetPut(s, k, v, r, t, d)</code>	<i>Asset-or-nothing put</i> Receive one unit of the asset if $s_t < k$ at expiration
<code>AssetDICall(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing down-and-in call</i> Receive one unit of the asset if $s_t > k$ at expiration and the barrier $H < s$ has been hit.
<code>AssetDOCall(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing down-and-out call</i> Receive one unit of the asset if $s_t > k$ at expiration and the barrier $H < s$ has not been hit.
<code>AssetDOPut(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing down-and-out put</i> Receive one unit of the asset if $s_t < k$ at expiration and the barrier $H < s$ has not been hit.
<code>AssetDIPut(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing down-and-out put</i> Receive one unit of the asset if $s_t < k$ at expiration and the barrier $H < s$ has been hit.
<code>AssetUICall(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing up-and-in call</i> Receive one unit of the asset if $s_t > k$ at expiration and the barrier $H > s$ has been hit.
<code>AssetUOCall(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing up-and-out call</i> Receive one unit of the asset if $s_t > k$ at expiration and the barrier $H > s$ has not been hit.
<code>AssetUOPut(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing up-and-out put</i> Receive one unit of the asset if $s_t < k$ at expiration and the barrier $H > s$ has not been hit.
<code>AssetUIPut(s, k, v, r, t, d, h)</code>	<i>Asset-or-nothing up-and-in put</i> Receive one unit of the asset if $s_t < k$ at expiration and the barrier $H > s$ has been hit.

4.3 Asian Options

There are straightforward closed-form solutions for geometric average European options, but not for arithmetic average options. Hence, the only functions implemented in Excel are for options with payoffs based on the geometric average price at expiration, G_t , which is computed over the life of the option.

The geometric average can be computed based on prices sampled N times over the life of the option. If N is not specified, a continuous average is assumed.

GeomAvgPriceCall(s, k, v, r, t, d, Optional N)	<i>Geometric average price call</i> Option with time t payoff $\max(0, G_t - k)$.
GeomAvgPricePut(s, k, v, r, t, d, Optional N)	<i>Geometric average price put</i> Option with time t payoff $\max(0, k - G_t)$.
GeomAvgStrikeCall(s, m, v, r, t, d, Optional N)	<i>Geometric average strike call</i> Option with time t payoff $\max(0, s_t - G_t)$.
GeomAvgStrikePut(s, m, v, r, t, d, Optional N)	<i>Geometric average strike put</i> Option with time t payoff $\max(0, G_t - s_t)$.