

Spreadsheet Option Functions Available with *Derivatives Markets*, second edition

Robert L. McDonald

August 29, 2005

Contents

1	Introduction	2
1.1	Spreadsheets	2
1.2	Using the Add-ins	2
1.3	A Note on Array Functions	3
2	Black-Scholes Functions	3
2.1	Prices	3
2.2	Greeks	4
2.3	Black-Scholes Array Functions	5
2.4	Black Formula	5
2.5	Perpetual American Options	5
2.6	CEV Pricing	6
2.7	Merton Jump Formula	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	9
4.2.2	Asset-or-Nothing	10
4.3	Asian Options	11
4.4	Compound Options	11
4.5	Exchange and Rainbow Options	12
5	Interest Rate Functions	12

1 Introduction

This is documentation for the option-pricing functions available in the spreadsheets accompanying the second edition of *Derivatives Markets*. All are user-defined functions in Excel, written in VBA, with the code accessible and modifiable via the Visual Basic editor built into Excel.

1.1 Spreadsheets

The following spreadsheets come with the book:

OptAll2.xls This provides examples of most (not all) of the pricing functions described here. The

OptAll2.xla This is an add-in version of OptAll2.xls. See Section 1.2.

OptBasic2.xls This provides the basic Black-Scholes and binomial calculations, along with formulas for barrier, binary, and perpetual options. It is a subset of OptAll2.xls.

OptBasic2.xla This is an add-in version of OptBasic2.

VBA_examples2.xls This contains the VBA examples from Appendix D.

Data.xls This spreadsheet provides stock price and option price data for some of the end-of-chapter problems.

1.2 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.

1.3 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 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>
BSCallImpVol(s, k, v, r, t, d, c)	<i>Implied volatility for European call</i> The option price is entered as “c”; the volatility entered does not matter.
BSPutImpVol(s, k, v, r, t, d, c)	<i>Implied volatility for European put</i> The option price is entered as “c”; the volatility entered does not matter.
BSCallImpS(s, k, v, r, t, d, c)	<i>Implied stock price for a given European call option price</i>
BSPutImpS(s, k, v, r, t, d, c)	<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>
<code>Black.Text_Greek(x)</code>	<i>Array function providing text description of the Greeks corresponding to items in the string “x”</i>

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.

<code>Function CallPerpetual(s, k, v, r, d)</code>	<i>Perpetual American call</i>
<code>Function PutPerpetual(s, k, v, r, d)</code>	<i>Perpetual American put</i>

2.6 CEV Pricing

The CEV model for the stock price specifies the instantaneous standard deviation of the stock return as

$$\sigma(S) = \bar{\sigma} S^{(\beta-2)/2} \quad (1)$$

Thus, the CEV pricing formula has two parameters in place of volatility: β and $\bar{\sigma}$. Setting these parameters appropriately requires care. If you observe a stock to have a 30% volatility, then given a β , you would set $\bar{\sigma}$ such that

$$\bar{\sigma} = 0.30 \times S^{(2-\beta)/2}$$

When $\beta = 2$, the CEV pricing model is the same as the Black-Scholes formula.

The CEV pricing functions are

<code>CEVCall(s,k,$\bar{\sigma}$,r,t,d,β)</code>	<i>CEV call price</i>
<code>CEVPut(s,k,$\bar{\sigma}$,r,t,d,β)</code>	<i>CEV put price</i>

2.7 Merton Jump Formula

The Merton formula for pricing with jumps requires three extra parameters: the jump probability, λ ; the expected size of a jump when one does occur, α_J ; and the volatility of the jump magnitude, σ_J . In the implementation here, the jump mean and volatility are measured with respect to the natural log of the jump.

The jump pricing formula returns both the call and put prices in a 2×2 array.

<code>MertonJump(s,k,v,r,t,d,λ,α_J,σ_J)</code>	<i>Call and put prices when the stock price can jump, with the jump magnitude determined by the log-normal distribution.</i>
---	--

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 lognormal tree is used (see CSection 11.3).

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)	<i>Binomial call</i>
BinomPut(s, k, v, r, t, d, opstyle, N)	<i>Binomial put</i>
BinomCallBermudan(s, k, v, r, t, d, opstyle, N, vest)	<i>Binomial Bermudan call</i>
BinomPutBermudan(s, k, v, r, t, d, opstyle, N, vest)	<i>Binomial Bermudan put</i>
BinomOptFixedDivCall(s, k, v, r, t, div_amt, div_h, div_next, opstyle, N)	<i>Binomial call price when under- lying asset pays dollar dividends in amount div_amt, with the first payment coming at time div_next, with dividends spaced div_h apart. The function</i>
BinomOptFixedDivPut(s, k, v, r, t, div_amt, div_h, div_next, opstyle, N)	<i>Binomial put price when under- lying asset pays dollar dividends in amount div_amt, with the first payment coming at time div_next, with dividends spaced div_h apart.</i>

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”.

CallDownIn(s, k, v, r, t, d, h)	<i>Down-and-in call</i>
CallDownOut(s, k, v, r, t, d, h)	<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

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

4.2.1 Cash-or-Nothing

$\text{CashCall}(s, k, v, r, t, d)$	<i>Cash-or-nothing call</i> Receive \$1 if $s_t > k$ at expiration
$\text{CashPut}(s, k, v, r, t, d)$	<i>Cash-or-nothing put</i> Receive \$1 if $s_t < k$ at expiration
$\text{CashDICall}(s, k, v, r, t, d, h)$	<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.
$\text{CashDOCall}(s, k, v, r, t, d, h)$	<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.
$\text{CashDOPut}(s, k, v, r, t, d, h)$	<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.
$\text{CashDIPut}(s, k, v, r, t, d, h)$	<i>Cash-or-nothing down-and-in put</i> Receive \$1 if $s_t < k$ at expiration and the barrier $H < s$ has been hit.
$\text{CashUICall}(s, k, v, r, t, d, h)$	<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.
$\text{CashUOCall}(s, k, v, r, t, d, h)$	<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.
$\text{CashUOPut}(s, k, v, r, t, d, h)$	<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 .

AssetCall(s, k, v, r, t, d)	<i>Asset-or-nothing call</i> Receive one unit of the asset if $s_t > k$ at expiration
AssetPut(s, k, v, r, t, d)	<i>Asset-or-nothing put</i> Receive one unit of the asset if $s_t < k$ at expiration
AssetDICall(s, k, v, r, t, d, h)	<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.
AssetDOWCall(s, k, v, r, t, d, h)	<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.
AssetDOPut(s, k, v, r, t, d, h)	<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.
AssetDIPut(s, k, v, r, t, d, h)	<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 (see Chapter 19). 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.

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

4.4 Compound Options

European compound options are options to buy or sell some other option, and as such have two expiration dates. The first, denoted t_1 , is the date at which the option to buy or sell an option must be exercised. The second, t_2 is the

date at which the bought or sold option expires. The strike price for buying or selling the underlying option at date $t1$ is denoted “x”.

CallOnCall(s, k, x, v, r, t1, t2, d)	<i>Compound call on call</i> Call option to buy a call option
PutOnCall(s, k, x, v, r, t1, t2, d)	<i>Compound put on call</i> Put option to sell a call option
CallOnPut(s, k, x, v, r, t1, t2, d)	<i>Compound call on put</i> Call option to buy a put option
PutOnPut(s, k, x, v, r, t1, t2, d)	<i>Compound put on put</i> Put option to sell a put option

4.5 Exchange and Rainbow Options

Exchange options and rainbow options have payoffs depending on two or more asset prices. An exchange option is the right to exchange one asset for another (ordinary calls and puts are exchange options where one of the two assets is cash). A rainbow option has the payoff $\max(S, Q, K)$, where S and Q are risky asset prices with dividend yields δ_s and δ_q , volatilities v_s and v_q , and correlation coefficient ρ . For binomial valuation, the number of binomial steps is N and the option style, OpStyle, is 0 for a European option and 1 for an American option.

BSCallExchange(s, v_s , δ_s , q, v_q , δ_q , ρ , t)	<i>European exchange call</i> Option to acquire s by giving up q .
BSPutExchange(s, v_s , δ_s , q, v_q , δ_q , ρ , t)	<i>European exchange put</i> Option to give up s in exchange for q .
BinomCallExchange(s, v_s , δ_s , q, v_q , δ_q , ρ , t, OpStyle, N)	<i>American exchange call</i> Option to acquire s by giving up q .
BinomPutExchange(s, v_s , δ_s , q, v_q , δ_q , ρ , t, OpStyle, N)	<i>American exchange put</i> Option to give up s in exchange for q .
RainbowCall(s, v_s , δ_s , q, v_q , δ_q , ρ , K, r, t)	<i>European rainbow call</i> Option on the maximum of s , q , and K .
RainbowPut(s, v_s , δ_s , q, v_q , δ_q , ρ , K, r, t)	<i>European rainbow put</i> Option on the minimum of s , q , and K .

5 Interest Rate Functions

These are array functions which compute prices, yields, delta, and gamma for zero-coupon bonds. These functions assume the short-term interest rate is generated by a process of the form

$$dr = a(b - r)dt + sr^\eta dZ$$

where $\eta = 0$ for the Vasicek formula and $\eta = 0.5$ for the CIR formula. The interest rate risk premium is ϕ .

Vasicek(a, b, ϕ , v, r, T)

Vasicek interest rate function Array function which returns the price of a zero-coupon bond paying \$1, the long-term yield, delta and gamma with respect to the interest rate, and the yield to maturity.

CIR(a, b, ϕ , v, r, T)

Cox-Ingersoll-Ross interest rate function Array function which returns the price of a zero-coupon bond paying \$1, the long-term yield, delta and gamma with respect to the interest rate, and the yield to maturity.