Chapter 1

OPTION PRICING WITH EXCEL*

Peter Honoré
Nyirendi Markets
Nyirendi Bank A/S
Klolehøj Str. 1-3
DK-2790 Copenhagen V, Denmark
phonor@dk

Rolf Poulsen
Department of Statistics and Operations Research
University of Copenhagen
Universitetsparken 5
DK-2100 Copenhagen 0, Denmark
rolfpou@dk

Abstract A closed-form solution is something you can program in Excel.

Keywords Excel, option pricing, binomial model, Black-Scholes model, partial differential equation, finite difference method, hedging.

1. Introduction

We show how to use spreadsheets for financial modelling, or more specifically: How to do option pricing with Excel and Visual Basic.

Most authors of papers in this volume demonstrate the use of their favorite program (carefully chosen from a long list of peers) in a field where it is less commonly used. This paper differs on two accounts. First, spreadsheets are already widely used in both the teaching of finance and in the finance industry - and have been for some time. So

*This file is peter@Home@home@paper.txt This version is dated August 17, 2001.

there is nothing particularly new there. Second, for all practical purposes 'spreadsheet' nowadays (2001) means 'Microsoft Excel' - and has done for some time. So there is no refined pro/con analysis of this vs. that spreadsheet.

You need an 'all-purpose-program'. One that has a battery of mathematical functions, is able to solve linear and not-so-difficult non-linear equations, has facilities for statistical analysis of data, can produce nice graphs, is endowed with a decent random number generator, offers the possibility of programming, especially a 'loop'-structure, and so on. Hundreds of software packages possess such features, but none is more widespread than the spreadsheet Excel that comes with the Microsoft Office package, and thus is on most PCs. Spreadsheets are typically quite accessible and offer a 'hand-on'-feeling: you change something and can see directly what happens. Also, spreadsheets go well conceptually with (discrete time, discrete space) financial models; the latter are true that you see on your screen is the model, not a particular outcome of some equations we have to solve. But spreadsheets also offer more advanced programming (like) features that allow us to deal with more complex models and analyze more sophisticated questions. Or at least help us perform initial analyses, that tell us what the interesting questions and the pitfalls are, if and when we take things further.

The rest of the paper goes as follows. In Section 2.1 we briefly review the principles of pricing by no arbitrage in binomial models, and show how this can be illustrated and implemented in Excel. Once we are confident with this, we do not need to see the whole model each and every time; we demonstrate how Excel's programming environment can be used to give us just what we want. This allows us to investigate the binomial model, especially concerning limiting behaviour, and leads us in Section 2.2 to consider the Black-Scholes model, which is a continuous-time model. Here there are two pricing methods: Monte Carlo simulation and numerical solution of partial differential equations (PDEs). When it can be used, the latter is typically the fastest, and we show how to implement a finite difference PDE solution method in Excel. Simulation is also important and in Section 2.3 we use it to study a topic that has recently received some attention: The effects of less-than-perfect hedging, especially the effects from discrete hedging and from model misspecification. In the concluding Section 3 we mention some other experiments that lack of space prevents from doing. Further, we have a short discussion of what we see as some of the advantages and disadvantages of spreadsheets in general and Excel in particular.
2. **Financial Calculations: Models, Problems, & Solutions**

2.1 The binomial model

The binomial model is the ‘workhorse’ when it comes to illustrating the principles of pricing by no arbitrage.\(^1\) The binomial model was first presented in Cox, Ross, and Rubinstein (1979) and the fundamental theorems of asset pricing first formulated in Harrison and Kreps (1979). They are now textbook material: Hull (1983) is the classic in the field, and Black (1997) also gives a nice albeit more abstract treatment.

The building block is the one-period model with two future states of nature. This model initially contains two assets that can be both bought and sold in any quantity by investors without transaction costs. Specifically, you can sell something you don’t have (known as ‘taking a short position’) if you just promise to honor your future obligations to the buyer. The first asset is a stock with an initial price of \(S(0) = s\). The future price of the stock is uncertain, stochastic; it can either (with probability \(p\)) go up to \(uS\), or it can go down to \(dS\), where \(u\) and \(d\) are real numbers that are known at time 0. The other asset is a risk-free one: a bank account where 1 invested $ grows to \(R\) no matter what. We assume that \(u > R > d\). Otherwise (if \(u\) and \(d\) are both greater than \(R\)) investors could just borrow money, buy the stock and obtain a risk-free profit after 1 period. (And if \(u\) and \(d\) are both smaller than \(R\), they could do the exact opposite.) This would be a ‘free lunch’, or an arbitrage opportunity. So what we are assuming is that the model is arbitrage-free.

Now suppose that we introduce into the economy a **European call option** on the stock with exercise, or strike, price \(K\) and maturity after one period. This is a contract that gives the owner the right but not the obligation to buy the stock after one period for the price \(K\). At time 1 the value of this call is equal to \(\max(uS - K, 0)\) (where \(\max(x, 0)\) means \(\max(x, 0)\)

\[
C(1) = \begin{cases} 
(uS - K)^+: & \text{up-state} \\
(ds - K)^+: & \text{down-state} 
\end{cases}
\]

The question is: What should the initial price of this call option be? A simple portfolio argument gives the answer: We can form a portfolio at time 0 using only the stock and the bank account which gives the same pay-off as the call at time 1 regardless of which state occurs. Let \((a, b)\) denote, respectively, the number of stocks and units of the bank account held at time 0 (so in this one-period model \(b\) is just how many $ you have deposited or borrowed in the bank). If the pay-off at time 1 is to match that of the call we must have

\[
au + br = cu \quad \text{and} \quad ad + br = cd.
\]

These two equations have the solution

\[
a = \frac{cu - cd}{s(u - d)}, \quad b = \frac{1}{R} \left(\frac{u - d}{u - d} - \frac{cu - cd}{s(u - d)}\right)
\]

and number of stocks to invest in is often suggestively written as \(a = \frac{C}{S}\) and is called the Delta hedge ratio, and we say that the portfolio hedges the call option. The cost of forming the portfolio \((a, b)\) at time 0 is \(aS(0) + b\). This is also the only possible initial price, say \(C(0)\), of the introduced call option. Any other price would create an arbitrage opportunity: If the price were lower, we could buy the call and sell the replicating portfolio \((a, b)\), receive cash now as a consequence and have no future obligations except to exercise the call if necessary. If the price were higher, the exact opposite could be done. So, \(C(0) = aS(0) + b\), which after some simple algebraic means:

\[
C(0) = \frac{(R - d)}{(u - d)} cu + \left(\frac{u - R}{u - d}\right) cd
\]

Now let \(q = \frac{R - d}{u - d}\) and note that the assumption of no arbitrage means that \(q \in [0, 1]\), so \(q\) can be interpreted as a probability and we can write the call price as

\[
C(0) = c_d \frac{u}{R} + (1 - q) c_u \frac{d}{R} = E_Q \left[ C(1) \right]
\]

i.e., an expected value using \(q\) as probability (indicated by the \(Q\) subscript) of the discounted time 1 value of the call. Note that the probability \(p\) plays no role in the expression for \(C(0)\).\(^2\) A stochastic process with the property that today’s value is the expectation of tomorrow’s value is called a martingale, and what we have shown in this example is that no arbitrage implies the existence of a set of probabilities such that the discounted call option price is a martingale if these probabilities are used. And it is not just for the call option that today’s value is the \(Q\) expected discounted value of the future price. It is also true for the bank account (trivially) and for the stock (just take \(K = 0\)). In fact we have shown that absence of arbitrage implies the existence of martingale probabilities \(q\), or in probabilistic terms an **equivalent martingale measure** (EMM) \(Q\). The converse is also true, and these two things combined is (a simple version of) what is known as the 1st fundamental theorem of asset pricing.\(^3\) "No arbitrage \(\implies\) EMM \(Q\)." Supposing
in 2 unknowns, which is typically impossible. And using a 2-point distribution as a model of stock prices a month, or a year, or ... from now on is not very realistic. It would seem that to impose completeness we need to assume that there are as many different assets as there are future states of the world. But there is a different, and much more realistic, way to remedy things. By piecing together a (large) number of simple one-period models and - and this is the ingenious insight - allowing for dynamic re-adjustments of portfolio, we can make models where we still only need the stock and bank account in the replicating portfolio. Let us look at this in a spreadsheet. The advantage is that you can both see the model and use the program to perform calculations. This is done in Figure 1.1 where we first encounter Excel. The upper panel shows a 2-period binomial model for the stock price. Today's price is 100 and at time 1 it can be either 107.79 or 92.77. Another coin is tossed at time 1 and the stock takes another move up or down (from the time-1 level). We have chosen a very specific parameterisation of the up- and down-moves, namely

\[ u = \frac{1}{d} = \exp(\sigma \sqrt{\Delta t}), \]

where the parameter \( \sigma \) is called the volatility, and \( \sigma^2 \) is the variance (per unit of time) of stock returns. This has the huge advantage that the stock price development can be described by a recombining tree or a lattice as 'up-down' sequence of moves back to the same price at a down-up-sequence. And now for the catch: If after one period we are in the up-state (cell B10), then there are only 2 possible future states and we can use our previous replication argument to conclude that a time-2 call option is worth 6.24 at time 1 if the stock goes up. Similarly, it is worth 0 if the stock goes down. Taking a step further back to time 0, we now know exactly which to possible time-1 values the call option can take. And the 1-period replication works again (and the arbitrage-free timed call option price is 3.48). Clearly, this backward recursive argument also works in 3-, 4-, or \( n \)-period models, and prices can be found by working backwards through the lattice.

The underlying mathematics is that the fundamental theorems of asset pricing still hold, i.e. in arbitrage-free models we have that arbitrage-free prices at time \( t \), \( \pi(t) \), are given by

\[ \pi(t) = E^Q \left[ \frac{S(t)}{S(0)} \right] \]

and if the equivalent martingale measure \( Q \) is unique then the model is complete, i.e. anything can be replicated.
Option Pricing with Excel

By piecing together many small models, we can get a more realistic model for the distribution of the future stock price. However, looking at — let alone generating by cut-and-paste — a full, say, 100-period lattice in a spreadsheet is cumbersome and impractical. Excel has an environment, Visual Basic (VBA), for programming macros and user-defined functions. To get to the VBA editor click on Tools → Macro → Visual Basic Editor in the Excel menu bar (or press Alt-F11). The commands are written in a so-called module (click on 'Insert → Module' in the VBA menu bar). The VBA editor also helps you debug the code, and you can then call it from the worksheet like any other function. The online help for VBA leaves something to be desired, so even though the syntax is fairly standard, a book such as Green, Bulen, and Martin (2000) is a must.

Thinking in pseudo-code terms, the back-end routine the pricing method is easy to program and we show explicit VBA code for the calculation of the price of a European call option in a binomial model. (Note that we do not have to represent the stock price grid as an object of size $(n + 1) \times (n + 1)$, but only $(n + 1) \times 2$, so we gain an order memory requirement.)

Option Explicit
Option Base 0

Function BinomialCall(As Double, q As Double, r As Double, T As Double, risk_free_rate As Double, stock_price As Double, delta(1 To 3) As Double, risk_free_rate As Double, r As Double, strike_price As Double, n As Integer, strike_price As Double) As Double
    Dim d As Double, d1 As Double, d2 As Double, v As Double, T As Double, n As Integer
    Dim grid(1 To 2, 1 To 2*n + 2) As Double
    Dim grid1 As Double, grid2 As Double
    Dim strike_price As Double

    For i = 1 To 2
        For j = 1 To 2*n + 2
            grid(i, j) = grid1*(1 + r) + grid2*(1 - r) + strike_price*(1 - q) / r
        Next j
        Next i

    BinomialCall = grid(1, 2*n + 2) + grid(2, 2*n + 2)

    Exit Function
End Function

Function BinomialPut(As Double, q As Double, r As Double, T As Double, risk_free_rate As Double, stock_price As Double, delta(1 To 3) As Double, risk_free_rate As Double, r As Double, strike_price As Double, n As Integer, strike_price As Double) As Double
    Dim d As Double, d1 As Double, d2 As Double, v As Double, T As Double, n As Integer
    Dim grid(1 To 2, 1 To 2*n + 2) As Double
    Dim grid1 As Double, grid2 As Double
    Dim strike_price As Double

    For i = 1 To 2
        For j = 1 To 2*n + 2
            grid(i, j) = grid1*(1 + r) + grid2*(1 - r) + strike_price*(1 - q) / r
        Next j
        Next i

    BinomialPut = grid(1, 2*n + 2) + grid(2, 2*n + 2)

    Exit Function
End Function}

This function can be called from Excel with valid arguments, e.g., 'BinomialPut(100, 0.15, 0.05, 105, 0.5, 100)'. Incidentally, these will be our standard parameters in the rest of the paper, i.e., if we don't tell you differently it is:

- The current stock price, $S_0$, is 100.
- The volatility, $\sigma$, is 0.15.
- The risk-free rate is 0.05, i.e., 5% p.a. (discretely or continuously compounded depending on the model).
- The strike price, $K$, is 105 (for both call and put).
- The maturity date, $T$, is 0.5.

With this code in place we can 'play around' and get a feeling for the model. One thing to do is to investigate the call option sensitivities to input parameters, current stock price, time to maturity, interest rate, volatility. . . These are known as ' Greeks', and given names as 'delta' (sensitivity to underlying), 'gamma' (delta-sensitivity to underlying), 'theta' (time-sensitivity), 'rho' (interest rate), 'vega' (volatility and incidentally not a Greek letter). The code can easily be extended such as to cover other payoff functions (e.g., $(K - x)^+$, i.e., the put option, American type features (i.e., the possibility of early exercise) or to return the delta rather than the price. We can also incorporate dividend payments from the stock, although this requires a bit more code, see Hull (1993) Ch. 14 [or instance]. Below we have modified the VBA code such that put options, possibly American type (American calls on non-dividend-paying stocks are never exercised prematurely; so they are not that interesting), can be priced, and the hedge-ratio (the number of stock we need to buy to hedge) can also be returned. The essential modification is that at each node it is checked whether we get more out of exercising immediately ($(S_n - K)^+$ for the call, $(K - S_n)^+$ for the put) than holding (the value of which is found as discounted (expectation of next period's outcomes), and exercising if that is the case.

Option Explicit
Option Base 0

Function BinomialPut(As Double, q As Double, r As Double, T As Double, risk_free_rate As Double, stock_price As Double, delta(1 To 3) As Double, risk_free_rate As Double, r As Double, strike_price As Double, n As Integer, strike_price As Double) As Double
    Dim d As Double, d1 As Double, d2 As Double, v As Double, T As Double, n As Integer
    Dim grid(1 To 2, 1 To 2*n + 2) As Double
    Dim grid1 As Double, grid2 As Double
    Dim strike_price As Double

    For i = 1 To 2
        For j = 1 To 2*n + 2
            grid(i, j) = grid1*(1 + r) + grid2*(1 - r) + strike_price*(1 - q) / r
        Next j
        Next i

    BinomialPut = grid(1, 2*n + 2) + grid(2, 2*n + 2)

    Exit Function
End Function

The function calculating option price based on a binomial model.

Option Explicit
Option Base 0

Function BinomialPut(As Double, q As Double, r As Double, T As Double, risk_free_rate As Double, stock_price As Double, delta(1 To 3) As Double, risk_free_rate As Double, r As Double, strike_price As Double, n As Integer, strike_price As Double) As Double
    Dim d As Double, d1 As Double, d2 As Double, v As Double, T As Double, n As Integer
    Dim grid(1 To 2, 1 To 2*n + 2) As Double
    Dim grid1 As Double, grid2 As Double
    Dim strike_price As Double

    For i = 1 To 2
        For j = 1 To 2*n + 2
            grid(i, j) = grid1*(1 + r) + grid2*(1 - r) + strike_price*(1 - q) / r
        Next j
        Next i

    BinomialPut = grid(1, 2*n + 2) + grid(2, 2*n + 2)

    Exit Function
End Function

The function calculating option price based on a binomial model.
Option Pricing with Excel

where the default values are the final possibilities.

Function bs(As Double, q As Double, r As Double, -
- As Double, T As Double, q As Double, r As Double, -
- As Double, S As Double, K As Double, D As Double) As Double

q = 1 - r / T
p = Exp(x / q)
q = q(x + q)
q = q(x / 0.5)
d = 1 / q

' Validate that q < 1 < d
If q > d Or q < d Then
call msgBox("Error in input data", 1)
End If

' Calculate the theoretical probability.
q = (e^(-<q>) / q = d)
IfOption grid(0) <> 0 Then
End If

' Call the binomial method.
For j = 0 To n
grid(j, p Right) = pg_eff(j + 1 - j) + (1 + j) * (1 - j) * K, q, optType)
Next j

' Call backward
For i = (n - 1) To 0 Step -1
For j = 0 To i
grid(i, k) = grid(i + 1 - k) + q * grid(i + 1 - k) + (i + k) / h:
Next j

' The row with options of the backward type.
For i = (n - 1) To 0 Step -1
For j = 0 To i
grid(i, k) = Application.Max(grid(i, k), pg_eff(i + (1 - j)) +
1 + j) * (1 - j) * K, q, optType)
Next j
Next i

' Choose the output result.
If Option(grid(Left)) Like "BS/Px"] Then
Exit Function
Else
' Default output is in the Price.
bin = grid(0)
End If
End Function

' Function for different payoff.
' Default call opt + pg_eff.
' Function pg_eff is Double, qk As Double, opt As String) As Double
'define the payoff function
pg_eff = OptionMax(K, 0)
Exit
' Others
pg_eff = OptionMax(K, 0)
End If
End Function

Figure 1.2. The graph shows the convergence of the call price to the Black-Scholes model where the number of periods, n, is high and step sizes are small. What happens in the extreme case, i.e., when n → ∞? In general it's hard to tell, but for the specific parameterization it is easy to investigate numerically, and in Figure 1.2 the result can be seen. It appears that call prices and hedge ratios converge, something that might be conjectured from the central limit theorem. This is true, in fact it can be proved that the whole model (in a certain sense) converges to the Black-Scholes model where the stock price is lognormal. This model we look at next. From Figure 1.2 we also note that the convergence is quite oscillatory, which is a potential source for problems, for instance if we want to extrapolate. There are different ways to improve this, see Kassler [2001].

2.2 The Black-Scholes model

We now turn to the types of computations we are likely to do when working with continuous-time financial models, a subject on which Björk...
Option Pricing with Excel

[1998], Musiela and Rutkowski [1997] and Duffie [1996] are all recommendable textbooks. Specifically, we look at the numerical solution of partial differential equations. A much more thorough treatment of this is given in Morton and Mayers [1994], and Wilmott [1998] shows how many, many problems in finance can be attacked and solved with a PDE approach.

In the Black-Scholes model the bank account, $\beta$, and develop as $\beta(t) = \exp(\rho t)$, and the stock price follows a Geometric Brownian motion,

$$dS(t) = \mu S(t)dt + \sigma S(t)dW$$

where $W$ is a Brownian motion under the 'real-world' probability measure $P$. The fundamental theorems of asset pricing still hold (at least with some extra technical requirements). Therefore 'arbitrage' means that the arbitrage-free prices at time $t$, $\pi(t)$, are given by

$$\pi(t) = E^Q_t\left(\text{discounted pay-off}\right)$$

where $Q$ is a probability measure such that the discounted stock price, $S/\beta$, is a martingale. A result from stochastic calculus (Girsanov’s theorem) tells us that the dynamics of $S$ must be

$$dS(t) = rS(t)dt + \sigma S(t)dW^Q,$$

where $W^Q$ is a $Q$-Brownian motion that is in some sense unique because the model is complete. The representation of prices as expected values suggests using simulation for calculating prices by appealing to the law of large numbers. This works for many different types of contracts (though not for American-type options without considerable research effort, see Fu, Laprise, Madan, Su and Wu [2001]) and is conceptually pretty straightforward. We will not look at that now. In the next section we will show how this comes to simulation in Excel, but use it in a more advanced context.

Rather, we focus on a more cunning approach that involves solving partial differential equations numerically. In the Black-Scholes model it can be shown (the Feynman-Kac representation) that a contract whose terminal pay-off depends only on the terminal stock price, say through the function $F$, has a time $t$ price of the form $\pi(t) = F(t, S(t))$, where $F$ is a deterministic function that solves the PDE,

$$F_t + rxF_x + \frac{1}{2}\sigma^2F_{xx} = rF$$

for $t < T$, with the terminal condition $F(T, x) = h(x)$. For the call option $h(x) = \max(x - K, 0)$, the PDE can be solved in closed form to give the Black-Scholes formula [Black and Scholes [1973]]

$$C(t) = F(S(t), t) = S(t)\Phi(z_t) - e^{-r(T-t)}K\Phi(z_t),$$

where $z_t = \left(\ln(S(t)/K) + (r + \frac{1}{2}\sigma^2)(T-t)\right)/\sigma\sqrt{T-t}$ and $\Phi$ is the cumulative density of the standard normal distribution.

**Finite Difference Methods**

Often it will not be possible to find closed-form solutions to the PDEs we encounter. So we need to look for numerical solution techniques, which is what we do next. (Naturally, the Black-Scholes formula will be our test-case.) To end this consider a PDE of the slightly more general form

$$F_t = LF$$

where $L$ is a differential operator that maps a function $f$ into the function $LFf$, where $LFf(x) = rxf(x) - a(x)f_x(x) - b(x)f(x)$, where $a(x)$ and $b(x)$ are known functions (and it is implicitly understood that $L$ on the right-hand side (RHS) of (1.2) works on the second argument of $F$). The pricing problem is to determine the time $0$ value of a contingent claim with final maturity $T$. The idea with finite difference is to discretize (1.2) by dividing the $x$ plane into a uniformly spaced mesh, $G$, with $N + 1$ discrete points in the time dimension and $M + 1$ points in the $x$ space:

$$x_j = x_0 + j\Delta x \quad \text{and} \quad t_j = j\Delta t$$

A probabilistic argument can be used for setting appropriate values for the lower and upper values of $x$. Set $x_0$ and $x_M$ such that a certain confidence interval of the state space in $x$ is reached given an initial value for $x$. Let $f_j$ denote the value of a function on the grid at node $(x_j, t_j)$. We then approximate the differential operators in (1.2) with difference operators on the grid. While $F$ is naturally approximated by $(f_{j+1} - f_j)/\Delta t$, there are several – non-equivalent; so it turns out – ways to approximate the operator $\partial^2 f/\partial x^2$ on a grid of degree of freedom' is to use the following on the interior of the grid:

$$\nu \approx (1 - \theta)\left(rf_j - \frac{1}{2}\sigma^2f_{j+1} - \frac{1}{2}\sigma^2f_{j-1} - 2f_j - f_{j\pm 1}\right)$$

$$+ \theta \left(rf_{j+1} - \frac{1}{2}\sigma^2f_{j+2} + f_{j+1} - 2f_{j+1} - f_{j+2}\right),$$

where $\theta$ is a parameter between 0 and 1. If $\theta = 1$ the resulting scheme is the so-called explicit (truncoidal) finite difference method, whereas we
have the fully implicit finite difference method for \( \theta = 0 \). The Crank-Nicolson scheme is an average of the first two schemes with \( \theta = 1/2 \). It is only when we use \( \theta \in (0, 1/2) \) that the numerical solution converges without further restrictions on the relation between time- and space-stepsizes. (We refrain from illustrating this in our examples.)

Let us consider the log-transformed Black-Scholes PDE, i.e., the PDE that the function defined by \( g(x) := F(e^x) \) solves. This means that \( a(x) = r - \sigma^2/2 \) and \( b(x) = \sigma \), and explains in part why we use the transform. We want as many things as possible to be constant. To solve the discretized system we get by plugging our approximations into 1.2 — and thus get an approximation to \( F \) and ultimately the timed-out contingent claim price — we have to solve a sequence of \( N \) systems of linear equations of the form

\[
A f^{j+1} = \phi^{j+1} \text{ for } j = 0, \ldots, N - 1,
\]

where \( A \) is an \((M + 1) \times (M + 1)\)-matrix, \( \phi^j = (f_0^j, f_1^j, \ldots, f_M^j)^T \) and the \((M + 1)\)-vector \( \phi^{j+1} \) depends only on quantities with time-index \( j + 1 \) (or greater). The matrix \( A \) is sparse; all its entries except those we mention below are 0. Calculations that are more tedious than hard give us the non-zero entries of \( A \) and the RHSs. We find

\[
a_{i,j-1} = \frac{1 - \theta}{2 \Delta x} \left( r - \sigma^2/2 - \frac{\sigma^2}{\Delta x} \right) = a_{i-1},
\]

\[
a_{i,j} = \frac{1}{\Delta t} \left( 1 - \theta \right) \left( 1 + \frac{\sigma^2}{\Delta x^2} \right) = a_0, \text{ and}
\]

\[
a_{i,j+1} = \frac{1 - \theta}{2 \Delta x} \left( -(r - \sigma^2/2) - \frac{\sigma^2}{\Delta x} \right) = a_i \text{ for } i = 1, \ldots, M - 1.
\]

The 'inner point' RHSs are given by

\[
\phi_i^{j+1} = -\frac{\theta}{\Delta t} \left( r - \sigma^2/2 - \frac{\sigma^2}{\Delta x} \right) f_i^{j+1} + \frac{1}{\Delta t} \left( 1 - \theta \right) \left( 1 + \frac{\sigma^2}{\Delta x^2} \right) f_i^{j+1}
\]

\[
+ \frac{\theta}{2 \Delta x} \left( r - \sigma^2/2 + \frac{\sigma^2}{\Delta x} \right) f_{i+1}^{j+1} \text{ for } i = 0, 1, \ldots, M - 1.
\]

It is not immediately obvious what we should do on the upper and lower boundaries. We suggest the following explicit boundary conditions. For the call option on the upper boundary use the approximation \( F^U = 1 \) (which in the log-transformed world means: \( g(X_M) = S_0 \exp(M \Delta x) \)) and \( F_{M+1} = 0 \). The reason for this is that if the stock price is high the \( \phi \) doesn't matter very much and then \( F(x) = e^{(r-\sigma^2/2)X} \).

\[K^t \Psi_S (s) = \epsilon^{-t} E^Q (S(T) - K) \Psi_S (s) = s - e^{-t \sigma^2/2} K.
\]

Similar arguments can be used at the low boundary and for put options, and in compact notation we end up with \( F_S (\text{any bd.}) = 0 \) and

\[F_S (\text{lower bd.}) = 1 (\text{lower bd.}), \quad F_S (\text{upper bd.}) = 1 (\text{upper bd.}).
\]

It is possible to use so-called implied boundary conditions (see Vetzal (1998)), where derivatives at the boundary are estimated using only positions on one side of the boundary. However, this makes the solution procedure unstable if drift does not dominate volatility (i.e., \( \sigma^2/\Delta x \) is large) at the boundaries. For the Black-Scholes model this is not the case.

All this means that with the explicit boundaries (for the call option) we have \( a_{i,0} = a_{i,M+1} = 0 \), while

\[
a_{i,1} = \frac{1}{\Delta t} \left( 1 - \theta \right) + r(1 - \theta),
\]

\[
\phi_i^{1} = \frac{1}{\Delta t} - r \Delta t \text{ and}
\]

\[
\phi_i^{0} = \frac{1}{\Delta t} - r \Delta t - (r - \sigma^2/2) S_0 \exp(M \Delta x).
\]

Since \( A \) is tridiagonal each of the systems in (1.3) can be solved with a computational effort that grows only linearly in the size of \( A \). Now we just have to roll back. Remember the final payoff of the contingent claim is the initial condition for \( \phi^0 \). Just as with the binomial model we can work with American-type options and other multi-payoff structures, as it is done in the following VBA code. (Where we do not explicitly show the algorithm, SolveLinearSystem, for solving tridiagonal linear systems, but refer the reader to Press, Teukolsky, Vetterling and Flannery (1992) for instance. It is possible to use the built-in Excel functions MINVERSE and MMULT for the matrix calculations. But this is much slower because the tridiagonal structure is not exploited.)
Figure 1.8. Price difference between American and European type put options.

\[ d(t) = \left( \frac{1}{d(t)} + \frac{\ln(x)}{d(t)} \right)^2 + \frac{\ln(x)}{d(t)} + \frac{\ln(x)}{d(t)} \]

\[ d(t) = \left( \frac{1}{d(t)} + \frac{\ln(x)}{d(t)} \right)^2 + \frac{\ln(x)}{d(t)} + \frac{\ln(x)}{d(t)} \]

\[ d(t) = \left( \frac{1}{d(t)} + \frac{\ln(x)}{d(t)} \right)^2 + \frac{\ln(x)}{d(t)} + \frac{\ln(x)}{d(t)} \]

The results of applying the PDE solver to American and European type put options can be seen in Figure 1.9.

Finite difference methods can be much faster than simulation, but are numerically more delicate. Small changes can make a big difference, as is shown in Figure 1.4. Here we smoothed the (otherwise non-differentiable) terminal condition using the Black-Scholes formulas in the next to last step and then working backwards from \( T - 2\Delta t \). So it is then true that we have a PDE solver with terminal condition \( (T - 2\Delta t, x, 0, x) \) at \( T - 2\Delta t \), and otherwise the same code. In this case 'the small change' works to our benefit, for instance because the smooth convergence makes extrapolation possible, but things might easily be the other way around. Another advantage of finite difference methods over simulation is that the partial derivatives are directly available. These are important for hedging purposes, as the next section will show.

2.3 Less-than-perfect Hedging

What makes pricing by arbitrage work is the possibility of creating perfectly replicating portfolios. This requires that you know the true model and its parameters with certainty, and in the Black-Scholes world that you can adjust your portfolio continuously. We now try to relax these assumptions, and in this way investigate the robustness of the Black-Scholes framework. A theoretical analysis of the topic is given in El Karoui, Jeanblanc-Picque and Shreve (1998), but we look at it in a simulation study. To this end we consider that case where we have sold a call option to someone, thus receiving some funds, but taking on a (possible) future liability. We want to \( \Delta \)-hedge, but are subject to certain constraints.

First, suppose that the Black-Scholes model is indeed the true one, but we are only able to adjust our portfolio discretely, say once every day or every week. Suppose that when we adjust, we make sure to keep the
Figure 1.1: Convergence of finite difference solutions to callprice PDEs for different \( N \). For the graph on the left, we used the "raw" payoff function as terminal condition, while we smoothed the payoff over the last time-step with the Black-Scholes formula to produce the graph on the right. This makes quite a difference: the convergence in the smoothed case is faster and more clear in the sense that it is monotone. Notice also the clearly superior accuracy of the Crank-Nicolson method (\( \theta = 1/2 \)) in the smoothed case.

number of steps prescribed by the theory, namely \( \Delta t = \Phi(z; (S_0, t_0)) \) and use the bank account to finance or deposit the cash-flows from the stock strategy. How bad are we off then? The following code can be used to answer that question.

```
Option Example
Option Base 0
Function BlackScholes(As Double, sigmas As Double, r As Double, K As Double, T As Double, = Sigmas = Double, polygon = Double, DiffEqSolve = Double) =
  Dim BackSolve As Double, Stock As Double, Multiplier As Double, DiffEqSolve = Double
  Dim BackSolve As Double, Stock As Double, Multiplier As Double, DiffEqSolve = Double
  Dim CallPrice As Double
  As Double
  Dim CallPrice As Double
  For j = 0 To Multiplier
    CallPrice = CallPrice + BackSolve = CallPrice
    For j = 0 To Multiplier
      CallPrice = CallPrice + BackSolve = CallPrice
      ' The call price
      Multiplier = DiffEqSolve(As Double, sigmas, r, K, T) - (1.0 - j, "Call_European", "Price")
      ' The call price
      Multiplier = DiffEqSolve(As Double, sigmas, r, K, T) - (1.0 - j, "Call_European", "Price")
      ' The call price
      Multiplier = DiffEqSolve(As Double, sigmas, r, K, T) - (1.0 - j, "Call_European", "Price")
    End For
  End For
  CallPrice = CallPrice + BackSolve = CallPrice
  CallPrice = CallPrice + BackSolve = CallPrice
  CallPrice = CallPrice + BackSolve = CallPrice
End Function
```

The output from this VB code is a range of size 2\( \times \)1. To use the function in a spreadsheet you have to make a 2\( \times \)1 range (by dragging with the left mouse-button down), enter the formula as you would normally, and finishing by pressing Ctrl+Shift+Enter (not just ENTER). When editing the formula later, it is also important to remember to use Ctrl+Shift+Enter to finish; Excel simply will not let you continue if you don't.

This experiment is 'point-by-numbers' proof that hedging works, and that it doesn't matter what the drift of the stock is. This is illustrated in Figure 1.5. In its left graph we have used a log-log scale to plot the dependence of the standard error of the hedging error (that is, the terminal value of our hedge portfolio less the pay-off of the call option) on the number of rebalancings for a stock price drift, \( \varepsilon_{r_0} \), of 0.07 (when the risk-free rate is 0.05). (The average hedging error is very close to 0, and doesn't depend very much on the number of rebalancings, so we haven't plotted it.) Not only do we see that the hedge becomes more and more accurate, but the fact that the points make up a straight line with slope \(-1/2\) indicates that the standard error of hedging error is proportional to \( 1/\sqrt{N} \) times we rebalance. (Indeed this is true, but in general it depends critically on the smoothness of the pay-off function, see Cleethorpe and Tenenbaum (2001)) The right graph in Figure 1.5 conveys another important message: the hedging error shows very little dependence on the stock price drift, \( \varepsilon_{r_0} \), and thereby hammers home the message of pricing by replication.

A variation of this experiment is to say that we adjust our hedge portfolio not every day, but only when our stock holdings are sufficiently far from the Black-Scholes \( \Delta \). This leads so-called Bandwidth hedging, see Wilmott (1998) (Ch. 21) and the references therein.

Another interesting question is: What happens if we hedge with a wrong volatility? In Figure 1.6 we can see what happens. For simplicity, we have \( \mu = r \), and the true volatility, \( \sigma_{\text{true}} \), is kept fixed while we have used different values for the hedging volatility, \( \sigma_{\text{hedge}} \), which is what we
think the volatility is and thus what we use in our hedge. The average hedging error is 0 exactly when we use the correct hedging volatility (it is easy to show that this must be so), while using \( \sigma_{\text{new}} > \sigma_{\text{true}} \) gives a positive average pay-off, which is reasonable, because we have sold the call option at a high price (higher than the arbitrage-free cost) and are getting the benefits from this. The standard error is averagely affected by both too low or too high hedging volatilities, but is fairly flat around the true volatility. All in all this would appear to be a case in favor just using a high hedging volatility, but two things should be kept in mind. First it may be that we can’t actually sell the call option for the price induced by our hedging volatility. And second, here we are trying to hedge a short position in a call meaning that our liabilities go up when the stock price go up. If we are trying to hedge a more complicated portfolio, this may not always hold. (More technically it depends on the sign of the second derivative of the position we’re trying to hedge w.r.t. our hedging instrument.)

3. Conclusion

"Sheer volume" means that Excel cannot be ignored. We showed how to use it for option pricing calculations. There are many relevant and interesting topics that we did not touch. On the finance side, we could have looked further into PDEs, investigated Black-Scholes-like AD-hedging (which is what is actually being done in practice) in the case where the stock price is not a Geometric Brownian motion, or discussed the issue of "realised modelling" when option prices observed in the market are used to create models with stock price dynamics that reproduce these option prices. This can all readily be done with Excel. On the Excel side, we didn’t show the available facilities for matrix calculations, data analysis or optimization. Nor did we talk about ‘real’ macros, as opposed to just user-defined functions; a function returns a value or a range calculated from inputs, whereas a macro can actively change various cells in the spreadsheet. So neither did we get around to the possibility of ‘embedding’ macros, that basically means that you show Excel what you want to do and it generates the code itself.

A further argument in favor of Excel is its strong interfacing opportunities (for example, Access or Reuter/Bloomberg financial services). And it shouldn’t be ignored that a number of very good "Excel-Finance-Books have recently appeared, for example, "Hedges and Moneyness" (2001) by Fox. Of course, the devil’s advocate would say that you should learn finance from a book on finance, not from a book on Excel (and vice versa).

Things we’re not so crazy about

As high-level software, spreadsheets are generally slow for numerical computations. The computations in this paper can all be done in “close to real-time” on a fairly new (mid-2001) PC, but they shouldn’t be much longer before we would recommend switching to something like C++ for the “number crunching”, make an ill of an object and then use Excel
REFERENCES


Notes

1. We refer the term 'hedging to an arbitrage' to 'hedging to an advantage', since the latter would give the impression that there is an exploitation of EBSPs among investors. There isn’t.
2. It is easily demonstrated that there is no relationship between $\pi$, $\gamma$ and the price of the stock today, but we don’t need it in the argument.
3. Even when regression is required there are many other obvious; in fact it has been a rather research topic in financial engineering over the last 20 or so years with cunning ways to construct the models. For instance such that the whole process of a model is often better, or such that theoretical concepts may be observed easier.
4. First, if you are certain you get a non-compressible model, which is a computational nightmare. Second, it is immovable gain, i.e., stock price + dividend discounted in the bank account, which should be O-series logon when discounted to present arbitrage.

References


