Spreadsheets in Education (eJSiE)

Volume 9 | Issue 3 Article 2

12-8-2016

Excel implementation of finite difference methods for option pricing

Timothy J. Kyng
Macquarie University, timothy.kyng@mq.edu.au

Sachi Purcal
Macquarie University, sachi.purcal@mq.edu.au

Jinhui C. Zhang
Macquarie University, colin.zhang@mq.edu.au

Excel implementation of finite difference methods for option pricing

Abstract
This paper presents and explains finite difference methods for pricing options and shows how these methods may be implemented in Excel. We cover both the explicit and the implicit finite difference methods. Each uses a numerical approximation to the partial differential equation and boundary condition to convert the differential equation to a difference equation. The difference equation can be solved using Excel and this solution is a numerical approximation to the option price. This paper explains how we obtain the difference equation from the differential equation and shows the reader how to implement and solve the difference equation using Excel.

Keywords
Option pricing, Numerical methods, Finite difference method, Implicit scheme, Explicit scheme, Excel implementation.

Cover Page Footnote
Corresponding author: timothy.kyng@mq.edu.au Department of Applied Finance and Actuarial Studies, Faculty of Business and Finance, Macquarie University, Sydney, NSW, 2109, Australia.

1 Introduction

Options are significant in the financial markets due to their use in financial product design, risk management applications, speculation, remuneration, and valuation of other securities. They are very actively traded by banks and other entities. There are many different types of option contract ranging from very simple to very complex.

Based on the foundations built in Black and Scholes (1973), the literature has proposed and implemented numerous models and methods for option pricing. There are many option contracts for which a closed form valuation formula is not available. In such circumstances we need to use numerical methods to solve the option pricing problem.

Cox, Ross and Rubinstein (1979) proposed a binomial tree method for option pricing. Generally the implementation of this binomial method tree is relatively easy to teach and for students to understand. It is readily implemented in `Excel`. Small scale examples can illustrate the major ideas of hedging, replication, risk neutral discounted expectation pricing and backwards recursion.

Following Cox, Ross and Rubinstein (1979), Boyle (1986) extended the binomial tree method to a trinomial tree method in which the stock price of each time period is assumed to have three outcomes: up, down and stable. With an additional outcome (stable) the trinomial tree method has higher accuracy than the binomial one.

Another widely used numerical method is the finite difference method. Indeed, the explicit finite difference method can be considered as a generalised version of the trinomial tree method. The implicit and explicit finite difference methods deal with the problem of computing the option price by approximating the partial differential equation by a difference equation and then numerically solving the difference equation. The binomial methods can be thought of as solving the problem using a discounted expectations approach via backwards recursion instead.

In our experience teaching actuarial and finance students about the binomial option pricing model is relatively easy, but teaching them about the finite difference method for option pricing is quite difficult. These students typically have no previous exposure to partial differential equations or software other than spreadsheets. Our purpose in writing this paper is to facilitate the teaching and learning of this part of option pricing theory. We believe that exposition of the topic via `Excel` is an excellent way to present the method and makes it much easier for students to

understand.

In this paper, we explain how to implement finite difference methods for option pricing using `Excel`. We explain the connection between the Black Scholes partial differential equation (PDE) and the finite difference methods for option pricing. The implicit and explicit finite difference approximations are examined in section 2 and in section 3 we discuss their implementation in `Excel` for European and American call and put options. Section 4 concludes.

2 Finite Difference Methods for Numerical Computation of Option Price

From Black and Scholes (1973), the Black-Scholes partial differential equation is

$$\frac{\partial F}{\partial t} + (r - y)\, S\frac{\partial F}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 F}{\partial S^2} - rF = 0 \tag{1}$$

with boundary conditions for the European call option of

$$F(S, T) = \max(S - K, 0), \tag{2}$$
$$F(0, t) = 0, \tag{3}$$

and, for $S \to \infty$,

$$F(S, t) = Se^{-y(T-t)} - Ke^{-r(T-t)}, \tag{4}$$

where the domain on which the function F is defined is $D = \{(S, t) : S \geq 0, 0 \leq t \leq T\}$, and F is the option price, t represents time, S is the stock price, K is the exercise price, y is the dividend yield, r is the risk free rate, σ is the volatility of the stock and T is the maturity date. The details of the proof can be found in Wilmott (2013) and Andreasen, Jensen and Poulsen (1998).

Solving the PDE analytically means finding $F(S, t)$ which satisfies both the PDE (1) and boundary conditions (2)–(4) on the domain D. Changing the boundary conditions will change the solution. Here we discuss the European call option. As we shall show below, for the call option boundary conditions the solution to the PDE is

$$F(S, t) = Se^{-y\tau}N(d_1) - Ke^{-r\tau}N(d_2), \tag{5}$$

where

$$d_1 = \frac{\ln S/K + (r - y + \sigma^2/2)\,\tau}{\sigma\sqrt{\tau}}, \tag{6}$$

$$d_2 = \frac{\ln S/K + (r - y - \sigma^2/2)\,\tau}{\sigma\sqrt{\tau}} \tag{7}$$

and $\tau = T - t$ is the time remaining till maturity as at time t; $N(d)$ is the cumulative normal density function.

If we change the boundary conditions to those for a European put option, which are

$$F(S,T) = \max(K - S, 0), \tag{8}$$

$$F(0,t) = Ke^{-r(T-t)} - Se^{-y(T-t)} \tag{9}$$

and, for $S \to \infty$

$$F(S,t) = 0, \tag{10}$$

then the solution to the PDE is

$$F(S,t) = Ke^{-r\tau}N(-d_2) - Se^{-y\tau}N(-d_1). \tag{11}$$

For many options (such as American options), there are no known analytic closed form solutions. In such cases we need to solve the PDE numerically. We address this problem in section 2.1 and the following.

To solve the PDE subject to its boundary conditions, we first convert it to another PDE we know how to solve by doing a set of transformations. This transformed PDE is in fact the heat equation from physics. It will have a boundary condition, too. We then solve this PDE using well known methods, namely a Green's function. We then reverse the transformations we did to get to the heat equation from the Black-Scholes equation, and this gives us our final solution, (5)–(7). The details of derivations can be found in Wilmott, Howison and Dewynne (1995).

Similarly, we can develop a closed form result for F as a put option price which is subject to boundary conditions (8)–(10).

As shown above, obtaining a closed form result is not straightforward. We have also noted above that for many options it is not possible to obtain a closed form

result. Thus, one must turn to numerical methods for a complete solution to the option pricing problem.

The class of numerical methods we focus on are known as finite difference methods. They use a finite difference approximation to the partial derivatives in the PDE. This converts the partial differential equation into a difference equation which we can solve numerically. It does this by using a discrete model of stock price, time and option value.

Before we pass on to a detailed discussion of this approach we must first set the scene for the numerical approach by adjusting the boundary conditions. For the call option, we adapt the definition of boundary condition (4) to apply for $S \geq S_{\max}$, whereas previously it applied for the more general $S \to \infty$. Similarly, for the put option, boundary condition (9) applies now for $S \leq S_{\min}$ while boundary condition (10) applies for $S \geq S_{\max}$.

Here, the term $S_{\max}$ means some value of S above which the call option is sufficiently deep in the money that its value converges to that of a long forward contract. Conversely, the term $S_{\min}$ means some value of S below which the put option is sufficiently deep in the money that its value converges to that of a short forward contract. As the stochastic process underlying the stock price is geometric Brownian motion, the probability distribution of the stock price at any time prior to maturity is lognormal. The extreme values ($S_{\max}$ and $S_{\min}$) of S need only be a few standard deviations above or below its mean for options to be deeply in or out of the money.

2.1 Finite Difference Approximations

Below we use the finite difference method to price an European put option . Hence we need to define the increments ΔS and ΔT in order to obtain the finite difference approximated form of the required partial derivatives.

Assume N equally spaced time intervals over the term of the option, T. Then ΔT, which is the length of each interval, is $\Delta T = T/N$. A European put option is deeply out of the money when the stock price S is extremely high. We assume there exists a stock price $S_{\max}$ such that, for $S \geq S_{\max}$, the put option is deeply out of the money with a value that is approximately zero, that is

$$S \geq S_{\max} \Rightarrow F(S,t) = 0. \tag{12}$$

Conversely, a European put option is deeply in the money when the stock price S is extremely low. We can assume there exists a low stock price $S \leq S_{\min}$ which makes the option deeply in the money and certain to be exercised at expiration. Hence, the option can be approximately regarded as a forward contract

$$S \leq S_{\min} \Rightarrow F(S,t) = Ke^{-r(T-t)} - Se^{-y(T-t)}.$$

Usually, we choose $S_{\min} = 0$, so that

$$S = S_{\min} \Rightarrow F(S,t) = Ke^{-r(T-t)}. \tag{13}$$

Breaking up the stock price into M equally spaced stock prices between $S_{\min}$ and $S_{\max}$, we obtain a stock price increment of $\Delta S = (S_{\max} - S_{\min})/M$. This allows us to create a grid like figure 1 of stock prices and times defined by an index i, which indexes the time, and an index j, which indexes the stock price level. We need to carefully choose the spacing in the stock price to ensure that one of the nodes coresponds to the current stock price. The range of values of i is $i = 0, 1, 2, \ldots, N$, and there are $N+1$ different values of i. The range of values of j is $j = 0, 1, 2, \ldots, M$.

The entry in row j and column i of the table is the stock price at time $i \times \Delta T$ after an increase of amount $j \times \Delta S$ in the stock price from the level $S_{\min}$. That is, the stock price is $S_{(i,j)} = S_{\min} + j \times \Delta S$ at row j and column i of the table. If we have $S_{\min} = 0$ then $S_{(i,j)} = j \times \Delta S$; we assume this hereafter.

We define the function $f(i,j) = F(j \times \Delta S, i \times \Delta T)$ as a discretised version of the function F. There are $(M+1) \times (N+1)$ different values of the function $f(i,j)$. The differential equation for the function $F(S,T)$ becomes a difference equation for $f(i,j)$ through the use of finite difference approximations to the partial derivatives in the PDE. We show the detailed derivation of the difference equation later.

In this paper, we demonstrate the finite difference method with a numerical example in `Excel`. All parameter values are shown in table 1. Suppose we have an European put option with certain parameters: initial stock price $S = \$30$, exercise price $K = \$30$ and term to maturity $T = 0.75$. For the grid, the number of time increments (steps) is $N = 3$ and the number of stock increments is $M = 6$. We also set the maximum stock price at $S_{\max} = 60$ and minimum stock price at $S_{\min} = 0$. Therefore the time increment is $\Delta T = T/N = 0.25$ and stock price increment is $\Delta S = (S_{\max} - S_{\min})/M = 10$. We also set the risk-free rate $r = 10\%$, the dividend yield rate to $y = 0\%$ and volatility $\sigma = 40\%$. We demonstrate our grid of stock prices in table 2.

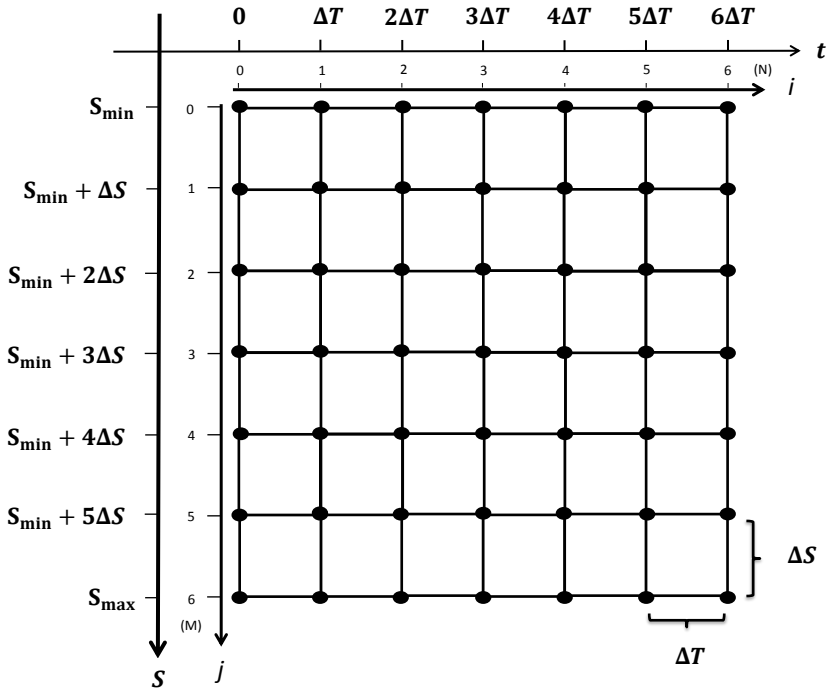

Figure 1: **The structure of the grid used in the finite difference approximation.** The horizontal axis represents time T, increasing from left to right in steps of ΔT. Each time step is indexed by i, which runs from 0 to its largest value of N. The vertical axis captures the stock price S, increasing from top to bottom in steps of ΔS from S_{min} to S_{max}. Each price level is indexed by j, which runs from 0 to its largest value of M. In this example both M and N are six, and if we were to summarise the graph in a table it would have seven rows $(M+1)$ and seven columns $(N+1)$.

Table 1: **Parameter values for the numerical example**

Symbol	**Meaning**	**Value**
S	Initial stock price (dollars)	30
K	Exercise value (dollars)	30
T	Term to maturity (years)	0.75
N	Number of time increments (steps)	3
S_{max}	Maximum stock price (dollars)	60
S_{min}	Minimum stock price (dollars)	0
M	Number of stock price increments	6
ΔT	Time increment (years)	0.25
ΔS	Stock price increment (dollars)	10
r	Risk free rate (continuously compounding annual rate)	0.1
y	Dividend yield (continuously compounding annual rate)	0
σ	Volatility (the standard deviation of the yearly logarithmic returns)	0.4

Table 2: **Stock prices $S_{(i,j)}$ at all nodes on the grid corresponding to the parameters in table 1.** Here $S_{(i,j)} = j \times \Delta S$ at time $i \times \Delta T$.

	$i = 0$	$i = 1$	$i = 2$	$i = 3$
$j = 0$	0.00	0.00	0.00	0.00
$j = 1$	10.00	10.00	10.00	10.00
$j = 2$	20.00	20.00	20.00	20.00
$j = 3$	30.00	30.00	30.00	30.00
$j = 4$	40.00	40.00	40.00	40.00
$j = 5$	50.00	50.00	50.00	50.00
$j = 6$	60.00	60.00	60.00	60.00

We now continue our numerical example by starting to build a table of $f(i,j)$. At a start, with boundary conditions: (8), (12) and (13), we can fill the numbers along the right hand edge, the top edge and the bottom edge of the table. This gives us table 3.

Table 3: **Values of $f(i,j)$ along the boundary of the grid.** Values at the right hand edge are given by the boundary condition at maturity, equation (8). At the bottom edge, use (10), modified by our discussion above to become (12). The top edge is given by (9), modified by our discussion above to become (13).

	$i=0$	$i=1$	$i=2$	$i=3$
$j=0$	27.83	28.54	29.26	30
$j=1$				20
$j=2$				10
$j=3$				0
$j=4$				0
$j=5$				0
$j=6$	0	0	0	0

2.2 Implicit Finite Difference Method

There are three common types of finite difference approximation to the derivative. A finite difference approximation to the derivative of F with respect to S at time $i \times \Delta T$ and with stock price $j \times \Delta S$ is $\partial F/\partial S \approx (f(i,j+1) - f(i,j))/\Delta S$. This is called a forward difference approximation. Another approximation is $\partial F/\partial S \approx (f(i,j) - f(i,j-1))/\Delta S$, which is known as a backward difference approximation. A third approximation is the average of the previous two,
$\partial F/\partial S \approx (f(i,j+1) - f(i,j-1))/2\Delta S$. Choosing to work with a particular approximation leads to a particular finite difference method, as we shall see below.

The derivative of F with respect to t at time $i \times \Delta T$ can be finite difference approximated as $\partial F/\partial t \approx (f(i+1,j) - f(i,j))/\Delta T$, which is a forward difference approximation.

For the second derivative of F with respect to S at time $i \times \Delta T$ and with stock price $j \times \Delta S$, the finite difference approximation can be written as $\partial^2 F/\partial S^2 \approx (f(i,j+1) + f(i,j-1) - 2f(i,j))/(\Delta S)^2$. We can derive this as follows. Since

the backwards difference approximation to $\partial F/\partial S$ at time $i \times \Delta T$, with stock price $j \times \Delta S$ and $(j+1) \times \Delta S$, are $\partial F/\partial S \approx (f(i,j) - f(i,j-1))/\Delta S$ and $\partial F/\partial S \approx (f(i,j+1) - f(i,j))/\Delta S$, the difference of these two approximations divided by ΔS is an approximation to $\partial^2 F/\partial S^2$. That is,

$$\frac{\partial^2 F}{\partial S^2} \approx \left(\frac{f(i,j+1) - f(i,j)}{\Delta S} - \frac{f(i,j) - f(i,j-1)}{\Delta S} \right) \times \frac{1}{\Delta S} \tag{14}$$

which, after some manipulation, yields the result stated at the beginning of the paragraph.

Now we make the following substitutions into the Black-Scholes PDE (1) and this will allow us to derive formulae for what is known as the implicit finite difference method. Taking the approximation to $\partial^2 F/\partial S^2$ given in the paragraph above, the 'average' approximation to $\partial F/\partial S$, the forward approximation to $\partial F/\partial t$, and recalling $f(i,j)$ is our discretised version of F, and substituting into (1) we obtain

$$\begin{aligned} 0 \;=\; & \frac{f(i+1,j) - f(i,j)}{\Delta T} \\ & + (r-y) \times (j\Delta S) \left(\frac{f(i,j+1) - f(i,j-1)}{2 \times \Delta S} \right) \\ & + \frac{1}{2}\sigma^2 (j\Delta S)^2 \left(\frac{f(i,j+1) + f(i,j-1) - 2f(i,j)}{(\Delta S)^2} \right) \\ & - rf(i,j). \end{aligned}$$

Then, the above equation can be rewritten as,

$$f(i,j-1) \cdot a(j) + f(i,j) \cdot b(j) + f(i,j+1) \cdot c(j) = f(i+1,j), \tag{15}$$

where the coefficients a, b and c are defined by

$$a(j) \equiv \frac{1}{2}(r-y) \times j\Delta T - \frac{1}{2}\Delta T \sigma^2 j^2, \tag{16}$$

$$b(j) \equiv 1 + \sigma^2 j^2 \Delta T + r\Delta T \tag{17}$$

and

$$c(j) \equiv -\frac{1}{2}(r-y) \times j\Delta T - \frac{1}{2}\Delta T \sigma^2 j^2 \tag{18}$$

for $i = 0, 1, 2, ..., N-1$ and $j = 0, 1, 2, ..., M-1$.

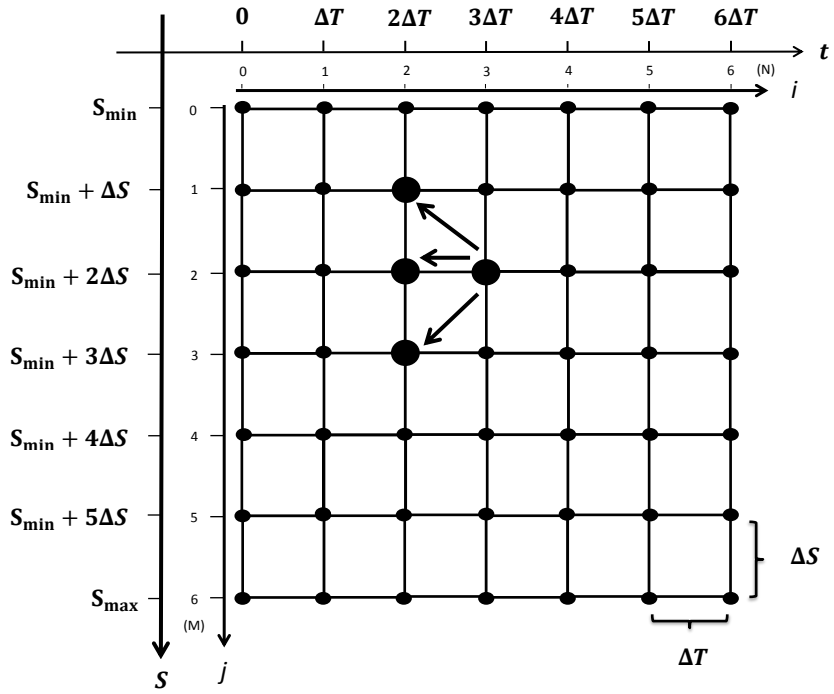

Figure 2: **Implicit finite difference method on the grid.** This is a graphical representation of equation (15). One sees the link between the option value at stock price $S = j\Delta S$ and time $t = (i+1)\Delta T$ with the nodes at time $t = i\Delta T$ and neighbouring stock prices. Our solution method proceeds backwards in time. As it does so, nodes at time $i\Delta T$ are determined by all the option values at time $(i+1)\Delta T$. Thus, values are not determined by their preceding neighbouring values, but in concert with all nodes at time $(i+1)\Delta T$, and so we term the method 'implicit'.

Equation (15) is represented diagrammatically in figure 2. There we see the link between a node at time $t = (i+1)\Delta T$ and stock price $S = j\Delta S$ with nodes at time $t = i\Delta T$ and stock prices $(j-1)\Delta S$, $j\Delta S$ and $(j+1)\Delta S$. As our solution process proceeds backwards through time (moving from right to left in figure 2), this approach is known as an implicit method—any node at time $i\Delta T$ is not directly determined by particular nodes at time $(i+1)\Delta T$, but rather in concert with all nodes at time $(i+1)\Delta T$. That is, at each time point, we have a set of simultaneous equations to solve, as we shall see below. This point will become clear as we detail the solution procedure below.

To price the option at $t = 0$, we first need to start with the values $f(i, j)$ for $i = N$ which are known, as these are the payoffs at maturity. At $i = N - 1$ (one time step before maturity) the boundary conditions (discussed in table 3) give us values at $j = 0$ and M. For the intermediate values $j = 1, 2, ..., M - 1$, the values

$f(N-1, j)$ are still unknown. To compute those values, we make use of the $M-1$ equations (15).

The numerical values of the coefficients a, b and c must first be calculated as functions of j, for which we take the values of the parameters r, y and σ from table 1. This yields the values of the coefficients a, b and c for the different values of j that appear in table 4. Note they vary by the stock price step on our solution grid; they do not vary by the time step.

Table 4: **Coefficients of equation** (15) **for our implicit finite difference example.** The coefficients are calculated using equations (16)–(18) and the parameters in table 1.

Stock price node	**Implicit example coefficients**		
j	$a(j)$	$b(j)$	$c(j)$
0	0.000 0	1.025 0	0.000 0
1	-0.007 5	1.065 0	-0.032 5
2	-0.055 0	1.185 0	-0.105 0
3	-0.142 5	1.385 0	-0.217 5
4	-0.270 0	1.665 0	-0.370 0
5	-0.437 5	2.025 0	-0.562 5
6	-0.645 0	2.465 0	-0.795 0

For this example, when $j = 2$ and substituting into equations (16)–(18), we have

$$\begin{aligned} a(j) &= \frac{1}{2}(0.10 - 0.00) \times 2 \times 0.25 - \frac{1}{2} \times 0.25 \times 0.40^2 \times 2^2 \\ &= 0.025 - 0.08 = -0.055 \\ b(j) &= 1 + 0.4^2 \times 2^2 \times 0.25 + 0.10 \times 0.25 = 1 + 0.16 + 0.025 = 1.185 \\ c(j) &= -\frac{1}{2}(0.10 - 0.00) \times 2 \times 0.25 - \frac{1}{2} \times 0.25 \times 0.40^2 \times 2^2 \\ &= -0.025 - 0.08 = -0.105 \end{aligned}$$

The calculations are tedious to do by hand, but very easy to do in a spreadsheet.

Our solution method proceeds recursively, moving backwards in time, from the highest time node values to the lowest. We set i in equation (15) to $N-1$, which for our grid with $N = 3$ and $M = 6$, is 2.

For this fixed $i = 2$, we must consider the possible values of j (excluding the boundaries), which range from 1 to 5. Thus, we have

$$\boxed{f(2,0)} \cdot a(1) + f(2,1) \cdot b(1) + f(2,2) \cdot c(1) = f(3,1), \tag{19}$$

$$f(2,1) \cdot a(2) + f(2,2) \cdot b(2) + f(2,3) \cdot c(2) = f(3,2), \tag{20}$$

$$f(2,2) \cdot a(3) + f(2,3) \cdot b(3) + f(2,4) \cdot c(3) = f(3,3), \tag{21}$$

$$f(2,3) \cdot a(4) + f(2,4) \cdot b(4) + f(2,5) \cdot c(4) = f(3,4) \tag{22}$$

and

$$f(2,4) \cdot a(5) + f(2,5) \cdot b(5) + \boxed{f(2,6)} \cdot c(5) = f(3,5). \tag{23}$$

We can rewrite the first equation (19) as

$$f(2,1) \cdot b(1) + f(2,2) \cdot c(1) = f(3,1) - f(2,0) \cdot a(1), \tag{24}$$

where the function values f on the RHS of (24) are known (boundary values we calculated previously) while those on its LHS are unknown. Equation (20) to (22) already fit this pattern; all that remains is to adjust the last equation (23) to

$$f(2,4) \times a(5) + f(2,5) \times b(5) = f(3,5) - f(2,6) \times c(5). \tag{25}$$

The above set of five equations (24), (20), (21), (22) and (25), i.e., the first and last modified equations and the original middle three from above can be written in matrix form as

$$\begin{pmatrix} b(1) & c(1) & 0 & 0 & 0 \\ a(2) & b(2) & c(2) & 0 & 0 \\ 0 & a(3) & b(3) & c(3) & 0 \\ 0 & 0 & a(4) & b(4) & c(4) \\ 0 & 0 & 0 & a(5) & b(5) \end{pmatrix} \times \begin{pmatrix} f(2,1) \\ f(2,2) \\ f(2,3) \\ f(2,4) \\ f(2,5) \end{pmatrix} = \begin{pmatrix} f(3,1) \\ f(3,2) \\ f(3,3) \\ f(3,4) \\ f(3,5) \end{pmatrix} - \begin{pmatrix} f(2,0)\, a(1) \\ 0 \\ 0 \\ 0 \\ f(2,6)\, c(5) \end{pmatrix},$$

which is a set of five simultaneous equation in five unknowns—the $f(2,\cdot)$ values are unknown, whereas the $f(3,\cdot)$, $f(2,0)$ and $f(2,6)$ values are known from our boundary conditions.

Rearranging, we can easily solve for these unknown values in `Excel`

$$\begin{pmatrix} f(2,1) \\ f(2,2) \\ f(2,3) \\ f(2,4) \\ f(2,5) \end{pmatrix} = \begin{pmatrix} b(1) & c(1) & 0 & 0 & 0 \\ a(2) & b(2) & c(2) & 0 & 0 \\ 0 & a(3) & b(3) & c(3) & 0 \\ 0 & 0 & a(4) & b(4) & c(4) \\ 0 & 0 & 0 & a(5) & b(5) \end{pmatrix}^{-1} \times \begin{pmatrix} f(3,1) - f(2,0)\, a(1) \\ f(3,2) \\ f(3,3) \\ f(3,4) \\ f(3,5) - f(2,6)\, c(5) \end{pmatrix}. \tag{26}$$

Equation (26) provides us with the values of the function f at time step $i = 2$ in terms of f values at time step $i = 3$. Indeed, in general, we can obtain the values of $f(i-1,\cdot)$ from the values of $f(i,\cdot)$. Therefore, the option values at the start of the grid, i.e., $f(0,\cdot)$, can be eventually calculated from the known maturity values, i.e., $f(T,\cdot)$. This step-by-step calculation process is known as backward recursion, or a time-marching approach.

For our particular numerical example in (26), the matrix[1]

$$\begin{pmatrix} b(1) & c(1) & 0 & 0 & 0 \\ a(2) & b(2) & c(2) & 0 & 0 \\ 0 & a(3) & b(3) & c(3) & 0 \\ 0 & 0 & a(4) & b(4) & c(4) \\ 0 & 0 & 0 & a(5) & b(5) \end{pmatrix}$$
$$= \begin{pmatrix} 1.065 & -0.0325 & 0 & 0 & 0 \\ -0.055 & 1.185 & -0.105 & 0 & 0 \\ 0 & -0.1425 & 1.385 & -0.2175 & 0 \\ 0 & 0 & -0.270 & 1.665 & -0.37 \\ 0 & 0 & 0 & -0.4375 & 2.025 \end{pmatrix}$$

is known, as is the vector

$$\begin{pmatrix} f(3,1) - f(2,0) \times a(1) \\ f(3,2) \\ f(3,3) \\ f(3,4) \\ f(3,5) - f(2,6) \times c(5) \end{pmatrix} = \begin{pmatrix} 20 - 29.24 \times -0.0075 \\ 20 \\ 10 \\ 0 \\ 0 - 0 \times -0.5625 \end{pmatrix} = \begin{pmatrix} 20.219 \\ 20 \\ 10 \\ 0 \\ 0 \end{pmatrix}.$$

We are then in a position to use `Excel` to solve equation (26), which yields the solution vector at time half a year after option issue as

$$\begin{pmatrix} f(2,1) \\ f(2,2) \\ f(2,3) \\ f(2,4) \\ f(2,5) \end{pmatrix} = \begin{pmatrix} 19.27 \\ 9.42 \\ 1.00 \\ 0.17 \\ 0.04 \end{pmatrix}.$$

[1]The matrix is tri-diagonal: it has non-zero entries along the main diagonal (from top left to bottom right) and non-zero entries along the diagonal above and the below this main diagonal, while has zero entries in all other positions.

Now that we have computed the values of the option at time step $i = 2$, the same approach can be applied to computing the values at time step $i = 1$ (from the values at time step $i = 2$).

In matrix form the equations to solve equation (15) for $i = 1$ are

$$\begin{pmatrix} b(1) & c(1) & 0 & 0 & 0 \\ a(2) & b(2) & c(2) & 0 & 0 \\ 0 & a(3) & b(3) & c(3) & 0 \\ 0 & 0 & a(4) & b(4) & c(4) \\ 0 & 0 & 0 & a(5) & b(5) \end{pmatrix} \times \begin{pmatrix} f(1,1) \\ f(1,2) \\ f(1,3) \\ f(1,4) \\ f(1,5) \end{pmatrix} = \begin{pmatrix} f(2,1) - f(1,0)\, a(1) \\ f(2,2) \\ f(2,3) \\ f(2,4) \\ f(2,5) - f(1,6)\, c(5) \end{pmatrix} \quad (27)$$

$$= \begin{pmatrix} f(2,1) \\ f(2,2) \\ f(2,3) \\ f(2,4) \\ f(2,5) \end{pmatrix} - \begin{pmatrix} f(1,0)\, a(1) \\ 0 \\ 0 \\ 0 \\ f(1,6)\, c(5) \end{pmatrix}.$$

The vector on the right hand side of equation (27) can be computed numerically, which is $(19.49, 9.42, 1.00, 0.17, 0.14)$, as we now have all the information we need. The solution, written in matrix notation, is then

$$\begin{pmatrix} f(1,1) \\ f(1,2) \\ f(1,3) \\ f(1,4) \\ f(1,5) \end{pmatrix} = \begin{pmatrix} 1.065 & -0.0325 & 0 & 0 & 0 \\ -0.055 & 1.185 & -0.105 & 0 & 0 \\ 0 & -0.1425 & 1.385 & -0.2175 & 0 \\ 0 & 0 & -0.270 & 1.665 & -0.37 \\ 0 & 0 & 0 & -0.4375 & 2.025 \end{pmatrix}^{-1} \times \begin{pmatrix} 19.49 \\ 9.42 \\ 1.00 \\ 0.17 \\ 0.04 \end{pmatrix}$$

and can be solved in `Excel` to give the values at time step $i = 1$ of

$$\begin{pmatrix} f(1,1) \\ f(1,2) \\ f(1,3) \\ f(1,4) \\ f(1,5) \end{pmatrix} = \begin{pmatrix} 18.57 \\ 8.96 \\ 1.70 \\ 0.40 \\ 0.10 \end{pmatrix}.$$

With the option values at time step $i = 1$, we can calculate the values of the option

at time step at time step $i = 0$ using the same approach, giving

$$\begin{pmatrix} f(0,1) \\ f(0,2) \\ f(0,3) \\ f(0,4) \\ f(0,5) \end{pmatrix} = \begin{pmatrix} 17.90 \\ 8.59 \\ 2.22 \\ 0.64 \\ 0.19 \end{pmatrix},$$

and it is $f(0,3)$, at a stock price of \$30, which we originally sought.

The above simplified numerical example explains the ideas behind the implicit finite difference method. We can formalise this solution method as follows: starting with a grid consisting of M increments in the stock price and N time increments, we calculate boundary conditions to give the top, bottom and rightmost values of the grid. Then, at each (backwards) time step we have a set of $M-1$ simultaneous equations to solve via a matrix approach

$$\mathbf{A} \times \mathbf{f}_i = (\mathbf{f}_{i+1} - \mathbf{d}_i), \tag{28}$$

where $\mathbf{f}_i = \begin{pmatrix} f(i,1) \\ f(i,2) \\ ... \\ f(i,M-2) \\ f(i,M-1) \end{pmatrix}$ and $\mathbf{d}_i = \begin{pmatrix} f(i,0)\, a(1) \\ 0 \\ ... \\ 0 \\ f(i,M)\, c(M-1) \end{pmatrix}$ are both vectors of dimension $M-1$ and

$$\mathbf{A} = \begin{pmatrix} b(1) & c(1) & 0 & 0 & 0 & ... & 0 \\ a(2) & b(2) & c(2) & 0 & 0 & ... & 0 \\ 0 & a(3) & b(3) & c(3) & 0 & ... & 0 \\ 0 & 0 & a(4) & b(4) & c(4) & ... & 0 \\ \vdots & \vdots & ... & ... & ... & ... & 0 \\ 0 & 0 & ... & a(M-2) & b(M-2) & c(M-2) & 0 \\ 0 & 0 & 0 & ... & 0 & a(M-1) & b(M-1) \end{pmatrix} \tag{29}$$

is an $(M-1) \times (M-1)$ tri-diagonal square matrix. The solution is $\mathbf{f}_i = \mathbf{A}^{-1} \times (\mathbf{f}_{i+1} - \mathbf{d}_i)$, which gives the vector of option values at time step i in terms of those at time step $i+1$. We apply this equation using backward recursion working through the $N-1$ backwards time steps from $i = N-1$ through to $i = 0$ to obtain the option price at time 0. It is possible to program this approach into `Excel`. To do so,

matrix calculations in `Excel` are required—in particular, matrix inversion, matrix multiplication, and addition and subtraction of vectors.

It can be shown that, under certain conditions, if we let the M and N get bigger then the above method will converge to the correct option value at time 0. We comment further on issues of stability and accuracy in section 2.4 below.

The matrix $\mathbf{A}$ in equation (28) is a tri-diagonal matrix and the solution $\mathbf{f}_i$ requires calculation of the inverse of this matrix. Fortunately, there is a very efficient algorithm for computing the solution of equations such as (28), known as the Thomas Algorithm. We can compute the solution easily using this algorithm and it can handle very large matrices. This is covered in section 2.4 of Press et al. (2007). However the algorithm is not built into `Excel` whereas matrix inversion is, and for purposes of exposition with `Excel` we have used matrix calculations instead of the Thomas algorithm.

2.3 Explicit Finite Difference Approach

The explicit finite difference method is an alternative to the implicit finite difference approach discussed above. It has the advantage that there are no simultaneous equations that need to be solved. Its disadvantage, though, is that, relative to the implicit finite difference method, it converges to a solution at a slower rate.

The explicit approach initially proceeds in much the same way as the implicit approach. We begin, however, with a different set of approximations to the partial derivatives that we chose for the implicit method, discussed above equation (14), specifically

$$\frac{\partial F}{\partial t} \approx \frac{f\left(i+1,j\right)-f\left(i,j\right)}{\Delta T}, \tag{30}$$

$$\frac{\partial F}{\partial S} \approx \frac{f\left(i+1,j+1\right)-f\left(i+1,j-1\right)}{2\cdot\Delta S} \tag{31}$$

and

$$\frac{\partial^2 F}{\partial S^2} \approx \frac{f\left(i+1,j+1\right)+f\left(i+1,j-1\right)-2f\left(i+1,j\right)}{\left(\Delta S\right)^2}. \tag{32}$$

Compared with the implicit approach, we use information at time $t+1$ rather than at time t to approximate the first and second derivatives of F with respect to S in the explicit approach, which could cause lower accuracy due to introduced errors.

Substitute (30)–(32) into the PDE of Black-Scholes model[2] (1):

$$\begin{aligned} 0 &= \frac{f(i+1,j)-f(i,j)}{\Delta T} \\ &\quad +(r-y)\cdot j\cdot \Delta S\frac{f(i+1,j+1)-f(i+1,j-1)}{2\cdot \Delta S} \\ &\quad +\frac{1}{2}\sigma^2 j^2\cdot(\Delta S)^2\,\frac{f(i+1,j+1)+f(i+1,j-1)-2f(i+1,j)}{(\Delta S)^2} \\ &\quad -r\cdot f(i,j). \end{aligned} \tag{33}$$

We can multiply (33) by ΔT throughout and collect together the terms involving $f(i+1,j-1)$, $f(i+1,j)$, $f(i+1,j+1)$ and $f(i,j)$ yielding

$$\begin{aligned} 0 &= f(i+1,j)-\sigma^2 j^2\Delta T\cdot f(i+1,j) \\ &\quad -f(i,j)-r\cdot f(i,j)\,\Delta T \\ &\quad +\frac{1}{2}(r-y)\cdot j\cdot \Delta T\cdot f(i+1,j+1)+\frac{1}{2}\sigma^2 j^2 f(i+1,j+1)\,\Delta T \\ &\quad -\frac{1}{2}(r-y)\cdot j\cdot \Delta T\cdot f(i+1,j-1)+\frac{1}{2}\sigma^2 j^2 f(i+1,j-1)\,\Delta T, \end{aligned}$$

which can be rewritten as

$$\begin{aligned} f(i,j) &= f(i+1,j-1)\times\left(\frac{-\frac{1}{2}(r-y)\cdot j\cdot \Delta T\cdot +\frac{1}{2}\sigma^2 j^2\Delta T}{1+r\cdot \Delta T}\right) \\ &\quad +f(i+1,j)\times\left(\frac{1-\sigma^2 j^2\Delta T}{1+r\cdot \Delta T}\right) \\ &\quad +f(i+1,j+1)\times\left(\frac{\frac{1}{2}(r-y)\cdot j\cdot \Delta T\cdot +\frac{1}{2}\sigma^2 j^2\Delta T}{1+r\cdot \Delta T}\right). \end{aligned} \tag{34}$$

Indeed, equation (34) can be re-expressed in a form similar to the implicit method equation (15), with

$$f(i,j)=f(i+1,j-1)\times a_j^*+f(i+1,j)\times b_j^*+f(i+1,j+1)\times c_j^*, \tag{35}$$

[2]Note that the $\Delta S,(\Delta S)^2$ terms in this expression cancel out.

where

$$a_j^* = \frac{-\frac{1}{2}(r-y)\cdot j\cdot \Delta T\cdot +\frac{1}{2}\sigma^2 j^2 \Delta T}{1+r\cdot\Delta T}$$
$$b_j^* = \frac{1-\sigma^2 j^2\Delta T}{1+r\cdot\Delta T}$$
$$c_j^* = \frac{\frac{1}{2}(r-y)\cdot j\cdot \Delta T\cdot +\frac{1}{2}\sigma^2 j^2 \Delta T}{1+r\cdot\Delta T}.$$

This is another backwards recursion formula and is represented diagrammatically in figure 3. Comparing figures 2 and 3 one can see it is different from the implicit finite difference method; it relates the value of the function at time step i with the three different values of the function f at time step $i+1$. This relationship means the method is easier to implement, as it does not require the solution of a set of simultaneous equations. That is, with the explicit finite difference method, we can proceed backwards from the terminal time f values to the initial time f values simply by use of the recursive equation (35).

2.4 Stability and Accuracy

The finite difference methods treated above are stable and accurate—under certain circumstances. Indeed, the remarkable Lax Equivalence Theorem tells us that for a well-posed PDE, a convergent numerical approximation scheme will get us to its true solution—as long as the scheme is stable (Duffy, 2006). These next paragraphs outline what is meant by each of these technical terms: well-posed, convergent numerical scheme and stability.

Initially one needs to determine whether the mathematical problem for which a solution is sought not only has a solution, but also whether that solution is "easy to find". Such a problem is known as well-posed, and these problems typically have a solution that changes little as one moves around its vicinity in 'small' steps (Lucchetti, 2006). Mathematicians term this "stable under small perturbations". That our option pricing problems are well-posed is intuitively obvious: the underlying economics tells us there is going to be only one price for the option contract at a particular point in time and this price will change smoothly in response to small changes in economic conditions.

A numerical scheme is known as convergent if, as the meshsize and the step sizes—or time-marching sizes—decrease, the finite difference scheme gets closer and closer

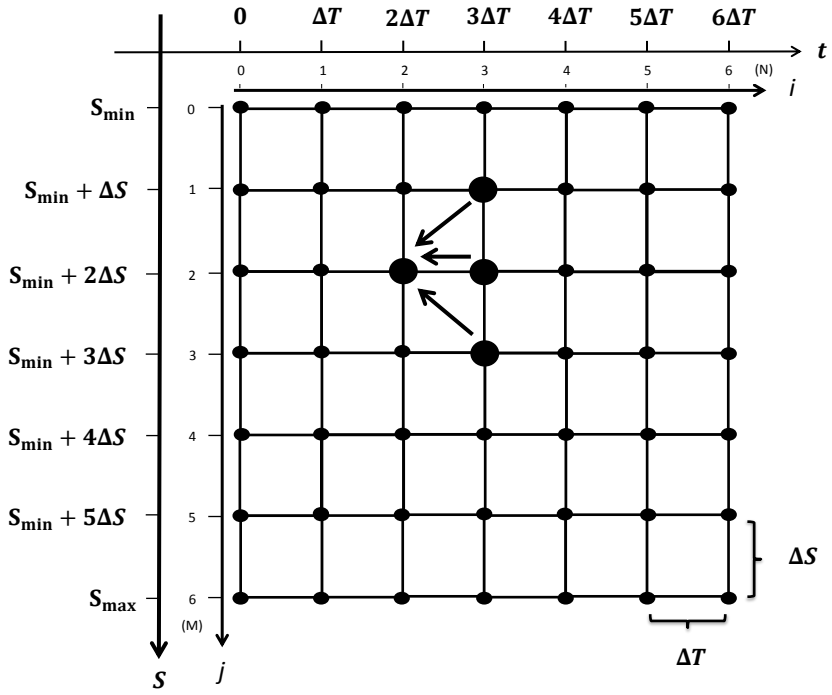

Figure 3: **Explicit finite difference method on the grid.** This is a graphical representation of equation (35). One sees the link between the option value at stock price $S = j\Delta S$ and time $t = i\Delta T$ with the nodes at time $t = (i+1)\Delta T$ and neighbouring stock prices. Our solution method proceeds backwards in time. As it does so, nodes at time $i\Delta T$ are determined by the option values at time $(i+1)\Delta t$. Thus, values are entirely determined by their preceding neighbouring values, and so we term the method explicit.

Table 5: **Percentage errors in pricing a European put option using the implicit finite difference scheme and parameter values from table 1, apart from M and N values, which vary.**

	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 200$
$M = 5$	22.5	23.1	23.3	23.4	23.7
$M = 10$	10.0	8.0	7.3	6.9	6.0
$M = 15$	0.8	0.8	1.2	1.6	2.3
$M = 20$	4.9	3.1	2.6	2.3	1.5
$M = 200$	3.4	1.7	1.1	0.9	0.1

to the differential equation it is trying to approximate (Shapira, 2006). Thus, we have chosen a good discretisation of the problem under consideration. This is also the case for the explicit and implicit finite difference schemes to determine option prices above. As step sizes get smaller our approximations to the derivatives get better and better. In each of tables 5, 6 and 7 we see, broadly, improved accuracy as M and N increase, reflecting the good convergence properties of the solution method. Further, we can observe parameter effects in tables 6 and 7. The situation with a more in-the-money option (table 6) has led to faster convergence. The situation with less market volatility (table 7), on the other hand, has led to slower convergence.

Table 6: **Percentage errors in pricing a European put option using the implicit finite difference scheme and parameter values from table 1, apart from M and N values, which vary, and S, which is set at 25 (the option is in-the-money).**

	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 200$
$M = 5$	4.6	4.6	4.6	4.6	4.6
$M = 10$	1.1	0.9	0.8	0.8	0.6
$M = 15$	0.6	0.9	1.0	1.1	1.2
$M = 20$	0.5	0.2	0.0	0.0	0.1
$M = 200$	0.7	0.4	0.3	0.2	0.0

Moving from the problem under consideration to focus on the approximation method we adopt, we find ideas of stability are also important. Key to any approximation method are its stability characteristics (Hackbusch, 2014). The stability

Table 7: **Percentage errors in pricing a European put option using the implicit finite difference scheme and parameter values from table 1, apart from M and N values, which vary, and σ, which is set at 0.1 (stock prices are less volatile than table 1 values).**

	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 200$
$M = 5$	664.4	660.6	659.4	658.7	657.0
$M = 10$	242.5	246.7	248.1	248.8	250.9
$M = 15$	76.8	72.3	70.7	69.9	67.8
$M = 20$	100.0	100.0	100.0	100.0	100.0
$M = 200$	2.4	1.5	1.2	1.1	0.8

characteristics of an implemented approximation method refer to the impact that small errors in the method have on results. If such small errors can produce big fluctuations in results—moving the approximate solution far away from the true solution—then the method has poor stability, and is of little value to us. The underlying mathematics of a finite difference scheme will suggest conditions that need to be satisfied for the scheme to be stable (John, 1982). Indeed, it can be shown that the explicit finite difference scheme is only stable if $0 < \Delta T/(\Delta S)^2 \leq 1/2$, while the implicit finite difference scheme is stable for any $\Delta T/(\Delta S)^2 > 0$ (Wilmott, Dewynne and Howison, 1993). Thus, stability and accuracy of these two finite difference methods requires these conditions being satisfied—for small enough values of ΔT and ΔS. And this is one of the reasons for the development of implicit finite difference methods. Such methods can achieve convergence without the efficiency loss implied by the extremely small time steps an explicit method may require for stability.

3 `Excel` Implementation

3.1 Implicit Finite Difference Method for Pricing an European Put and Call Option and an American Put Option

We firstly illustrate the `Excel` implementation of the implicit finite difference method for a European put option.

	A	B
1	**Implicit finite difference method spreadsheet**	
2		
3	**Valuation of a European put option**	
4		
5	**Table 1: Parameter values for the numerical example**	
6	30	S=initial stock price
7	30	X=exercise price for put
8	0.75	T=term to maturity
9	3	N=number of time increments (steps)
10	6	M=number of stock price increments
11	60	Smax=maximum stock price
12	0	Smin=minimum stock price
13	60	Range=Smax-Smin
14	10	Delta S
15	0.25	Delta T
16	0.1	risk free interest rate
17	0	dividend yield

Figure 4: **Parameters used in the `Excel` implementation of the implicit finite difference method for a European put option.** These are a subset of the values given in table 1 of section 2.1 above.

The parameters for our implementation of the implicit finite difference method are set out in the cell range A6:A17 of our spreadsheet, as shown in figure 4 below. We are using $N = 3$ time increments and $M = 6$ stock price increments. The stock price increment is \$10 and the stock prices range from \$0 to \$60. The time increment is 0.25 years and the times range from 0.00 to 0.75 years, going up in these steps of size 0.25 years. Using the closed form solution of the Black-Scholes model from (11) for this European put option, we found the option has a value of \$2.98.

Following the steps outlined in Section 2.1, we need to create a table for stock prices and a table for the option values at these different stock prices. Those `Excel` tables are indexed by a column index i for time and a row index j for price. In the spreadsheet we create `Excel` table 2 for the stock price and `Excel` table 3 for the option values for boundary conditions: (8), (12) and (13), which are shown in figure 5.

	D	E	F	G	H	I	J	K
3	**Table 2: Stock price S(i,j) at all nodes on the grid corresponding to the parameters**							
4								
5								
6			**0**	**1**	**2**	**3**		
7		**0**	0.00	0.00	0.00	0.00		
8		**1**	10.00	10.00	10.00	10.00		
9		**2**	20.00	20.00	20.00	20.00		
10		**3**	30.00	30.00	30.00	30.00		
11		**4**	40.00	40.00	40.00	40.00		
12		**5**	50.00	50.00	50.00	50.00		
13		**6**	60.00	60.00	60.00	60.00		
14								
15	**Table 3: Values of f(i,j) along the boundary of the grid**							
16								
17								
18			**0**	**1**	**2**	**3**		
19		**0**	27.8323	28.53688	29.2593	30		
20		**1**				20		
21		**2**				10		
22		**3**				0		
23		**4**				0		
24		**5**				0		
25		**6**	0	0	0	0		

Figure 5: **Stock prices and boundary conditions produced by our modelling.** The `Excel` tables reflect the modelling done in tables 2 and 3 of section 2.1 above.

For `Excel` table 2, the `Excel` code in cell F7 is =$A$12+$E7*$A$14 and we can copy this cell to F7:I13 to produce the stock prices. For `Excel` table 3, the `Excel` code in cell I19 is =MAX($A$7-I7,0) and we can copy this cell to I19:I25 for the option values using boundary condition (8). Then we input =$A$7*EXP(-($A$9-F$18)*$A$15*$A$16)-F7*EXP(-($A$9-F$18)*$A$15*$A$17) in F19. We can then copy that to the range F19:H19 to compute the option values using boundary condition (13). In cells F25:H25, we input number zero in for the option values in accordance with boundary condition (12).

In our spreadsheet we set up `Excel` tables 4 and 5 for the computation of the coefficients $a(j)$, $b(j)$ and $c(j)$, from (15), and the tri-diagonal matrix $\mathbf{A}$, from (29), which are shown in figure 6. The `Excel` code in cell F31 is =0.5*($A$16-$A$17)*E31*$A$15-0.5*$A$18^2*E31^2*$A$15 and we copy cell it to F32:F37 to compute all $a(j)$ coefficients. The `Excel` code in cell G31 is =1+$A$18^2*E31^2*$A$15

	D	E	F	G	H	I	J
27	**Table 4: Coefficients for our implicit finite difference example**						
28							
29							
30			a(j)	b(j)	c(j)		
31		**0**	0.0000	1.0250	0.0000		
32		**1**	-0.0075	1.0650	-0.0325		
33		**2**	-0.0550	1.1850	-0.1050		
34		**3**	-0.1425	1.3850	-0.2175		
35		**4**	-0.2700	1.6650	-0.3700		
36		**5**	-0.4375	2.0250	-0.5625		
37		**6**	-0.6450	2.4650	-0.7950		
38							
39	**Table 5: Tri-diagonal matrix, A**						
40							
41			**1**	**2**	**3**	**4**	**5**
42		**1**	1.0650	-0.0325	0.0000	0.0000	0.0000
43		**2**	-0.0550	1.1850	-0.1050	0.0000	0.0000
44		**3**	0.0000	-0.1425	1.3850	-0.2175	0.0000
45		**4**	0.0000	0.0000	-0.2700	1.6650	-0.3700
46		**5**	0.0000	0.0000	0.0000	-0.4375	2.0250
47							

Figure 6: **The coefficients in the difference equation and the tri-diagonal matrix.** These are produced by our modelling in tables 4 of section 2.2 and subsequent discussion. The coefficients are given by equations (16)–(18) above; matrix **A** by equation (29).

+$A$16*$A$15 and we copy it to G32:G37 to compute all $b(j)$ coefficients. The `Excel` code in cell H31 is =-0.5*($A$16-$A$17)*E31*$A$15-0.5*$A$18^2*E31^2*$A$15 and we can copy it to H32:H37 to compute all $c(j)$ coefficients. In `Excel` table 4, we input coefficient values from `Excel` table 5 based on (29). The numbers in bold on the top edge of table 5 are the i values, the column index, and the numbers in bold on the left edge are the j values, the row index, for the matrix.

Now we start to compute the option prices using backward recursion and the matrix formula $\mathbf{f}_i = \mathbf{A}^{-1} \times (\mathbf{f}_{i+1} - \mathbf{d}_i)$. Following the definition of the $\mathbf{d}_i$, given after equation (28), the adjustment vectors are calculated in `Excel` table 6—see figure 7 below. In the cell F50, we input the `Excel` code =F19*$F$32 and we can copy it to G50:H50 to compute the first entries in each of the other adjustment vectors. Cell F54 contains the `Excel` code =F25*$H$36; copy it to G54:H54.

`Excel` table 7, shown in figure 7, gives the option values at the interior points of the grid. Firstly, we select the cell range range H59:H63 and enter code for the matrix

	D	E	F	G	H	I
48	**Table 6: Adjustment vector, d_i**					
49			**0**	**1**	**2**	
50		**1**	-0.2087	-0.2140	-0.2194	
51		**2**	0.0000	0.0000	0.0000	
52		**3**	0.0000	0.0000	0.0000	
53		**4**	0.0000	0.0000	0.0000	
54		**5**	0.0000	0.0000	0.0000	
55						
56						
57	**Table7: Option prices**					
58			**0**	**1**	**2**	**3**
59		**1**	17.90	18.57	19.27	20.00
60		**2**	8.59	8.96	9.42	10.00
61		**3**	**2.22**	1.70	1.00	0.00
62		**4**	0.64	0.40	0.17	0.00
63		**5**	0.19	0.10	0.04	0.00
64						

Figure 7: **Tables of adjustment vectors $\mathbf{d}_i$ and option value vectors $\mathbf{f}_i$.** Details of these vectors appear immediately after equation (28). The bold entry is $f(0,3)$, the European put option price given by this toy implicit scheme at initiation if the stock price at that time is \$30, which is what we sought.

calculation =MMULT(MINVERSE(\$F\$42:\$J\$46),(I59:I63-H50:H54)). Recall that \$F\$42:\$J\$46 is our tri-diagonal matrix $\mathbf{A}$, appearing in figure 6 above and defined in equation (29). Then we hold down both the control and the shift keys while pressing the enter key to finish the formula input for this matrix. The combination of the control, shift and enter keys is required to create a matrix formula in `Excel`. This code implements the calculation for the case $i = 2$, which is one time step before maturity. We copy the code for the matrix calculation to G59:G63 and repeat the previous step of pressing these three keys simultaneously to compute the option price vector at time 1. Finally, we copy the code for the matrix calculation to F59:F63 and repeat the previous step to obtain option values at time 0. Implementing this in `Excel` is cumbersome for large M and N and in industrial practice the finite difference methods would be implemented in some other software package, using the Thomas Algorithm for solving equation (28) as explained above.

We find the option price we want in the third entry of the option price vector

at time 0. Using the implicit finite difference method, we calculate option price of 2.22. Compared to the analytic solution from the Black-Scholes model, 2.98, our finite difference implementation with $N = 3$ and $M = 6$ is not very accurate. The ratio of the finite difference method price to the analytic price is 74.24%.

However, here we only used a small-sized grid as a demonstration. If we repeat the method with higher values of N and M, we get more accurate results. Using $N = 8$, $M = 10$, we get a European put option price of 2.73 which is 91.53% of the analytic solution. With even greater grid size, $N = 25$, $M = 30$, we obtain the European put option value of 2.95 which has an accuracy ratio of 98.75%.

It is easy to modify the code for pricing a **European call option**. So far we have set up the `Excel` code for pricing a European put option with $N = 3$ and $M = 6$. We need to change the boundary conditions (2)–(4) in `Excel` table 3 to those of a European call option. Modified `Excel` tables 3, 6 and 7 are shown in figure 8.

`Excel` tables 2, 4 and 5 do not change—the `Excel` code is same for those tables as for the put option. The call option value is \$4.36 using the finite difference method, but \$5.15 using the analytic formula.

Lastly, using the same parameters, we use the `Excel` spreadsheet to price an **American style put option**. Most of the code and tables we have developed for the European put option can be reused for this purpose. Changes are made in `Excel` table 7, shown in figure 9. The `Excel` code in cell range F59:M63 is different now, as it takes account of the possibility of early exercise. The code in cell F59 is =MAX(INDEX(MMULT(MINVERSE(\$F\$42:\$J\$46),(G\$59:G\$63-F\$50:F\$54)),\$E59), MAX(\$A\$7-F8,0)).

The INDEX function picks out the correct element of the vector of option prices from the matrix calculation. The MAX function chooses the higher of either the early exercise value or the value assuming we do not exercise early. This produces a higher overall valuation. We then copy cell F59 to F59:H63 to compute all the other option values in the interior of the grid. The numerical results are, typically, greater than our results for the European put.

	D	E	F	G	H	I
15	**Table 3: Values of f(i,j) along the boundary of the grid**					
16						
17						
18			**0**	**1**	**2**	**3**
19		**0**	0.0000	0.0000	0.0000	0.0000
20		**1**				0.0000
21		**2**				0.0000
22		**3**				0.0000
23		**4**				10.0000
24		**5**				20.0000
25		**6**	32.1677	31.4631	30.7407	30.0000
26						
48	**Table 6: Adjustment vector, d_i**					
49			**0**	**1**	**2**	
50		**1**	0.0000	0.0000	0.0000	
51		**2**	0.0000	0.0000	0.0000	
52		**3**	0.0000	0.0000	0.0000	
53		**4**	0.0000	0.0000	0.0000	
54		**5**	-18.0943	-17.6980	-17.2916	
55						
56						
57	**Table7: Option prices**					
58			**0**	**1**	**2**	**3**
59		**1**	0.04	0.02	0.00	0.00
60		**2**	0.73	0.41	0.15	0.00
61		**3**	**4.36**	3.15	1.73	0.00
62		**4**	12.79	11.85	10.90	10.00
63		**5**	22.34	21.56	20.77	20.00

Figure 8: **Implicit finite difference method ($N = 3$, $M = 6$) for a European call option**. To switch from a put to call option we have to modify the values of $f(i, j)$ along the boundary of the grid to reflect the call option boundary conditions given in equations (2)–(4) in section 2 above. Then we follow through the implicit scheme methodology with this revised grid boundary. The bold entry is $f(0, 3)$, the call option price at contract initiation if the stock price, at that time, is \$30.

	D	E	F	G	H	I
57	**Table7: Option prices**					
58			**0**	**1**	**2**	**3**
59		**1**	20.00	20.00	20.00	20.00
60		**2**	10.00	10.00	10.00	10.00
61		**3**	2.36	1.76	1.00	0.00
62		**4**	0.67	0.41	0.17	0.00
63		**5**	0.20	0.11	0.04	0.00
64						

Figure 9: **Implicit finite difference method ($N = 3$, $M = 6$) for a American put option**. Here the cell formulæ of figure 7 above are altered to allow for the possibility of early exercise. Compared to the European put option price results in figure 7, we see the American put option prices are never smaller, and often larger.

3.2 Explicit Finite Difference Method for Pricing an European Put Option and an American Put Option

The `Excel` implementation of the explicit finite difference method is easier—it does not require manipulations of a tri-diagonal matrix, as can be seen from difference equation (35). We illustrate the way to use the explicit method to price a European put option below. The parameters for the valuation are in cell range A4:B16 and the coefficients $a^*(j)$, $b^*(j)$ and $c^*(j)$ are in cell range E4:H15 in figure 10.

In figure 10, the `Excel` code in cells F5, G5 and H5 is
=(-0.5*$B$11*$B$10*$E5+0.5*$B12^2*B$10*$E5^2)/(1+$B$11*$B$10),
=1/(1+$B$11*$B$10)*(1-$B$12^2*$E5^2*$B$10) and =(0.5*$B$11*$B$10*$E5
+0.5*$B$12^2*$B$10*$E5^2)/(1+$B$11*$B$10), respectively. We copy F5:H5 to F6:H15.

The grid of option prices and the boundary conditions is shown in figure 11. In A23:B28 we show the details of the analytic Black-Scholes valuation of the European put option. The details of the explicit finite difference method calculations of the option price are in the cell range D19:N32.

Cell range F20:N20 shows the time steps and the cell range F21:N21 shows the time remaining to maturity for these time steps. The cell range D22:D32 shows the values of the (price) index j and the cell range E22:E32 show the values of the stock price corresponding to these index values. From boundary condition (12), we

	A	B	C	D	E	F	G	H
1	**Explicit finite difference method example**							
2	**European put option**							
3								
4	S	20			**j**	**a(j)**	**b(j)**	**c(j)**
5	K	21			0	0.000	0.996	0.000
6	Smax	40			1	0.001	0.989	0.005
7	Smin	0			2	0.009	0.969	0.017
8	ΔS	4			3	0.024	0.936	0.036
9	T	0.3333			4	0.045	0.890	0.061
10	ΔT	0.0417			5	0.073	0.830	0.093
11	R	10.00%			6	0.107	0.757	0.132
12	σ	40.00%			7	0.148	0.671	0.177
13	y	0%			8	0.196	0.571	0.229
14	# steps				9	0.250	0.458	0.288
15	N	8			10	0.311	0.332	0.353
16	M	10						

Figure 10: **Parameter and coefficient values used in applying the explicit finite difference method to pricing a European put option**. Note that our explicit method example uses different parameter values to the implicit method example above. The mathematical detail of coefficients $a^*(j)$, $b^*(j)$ and $c^*(j)$ is given in the discussion immediately following equation (35) above.

put 0 in F32:M32. With boundary condition (8) at maturity, we put =MAX($B$5-$D22*$B$8,0) in cell N22 and copy it to N23:N32. Based on boundary condition (13), we put =$B$5*EXP(-($B$15-F$20)*$B$10*$B$11)-$E22*EXP(-($B$15-F$20)*$B$10*$B$13) in F22 and copy it to G22:M22.

To compute the option values in F23:M31, we enter the formula =MMULT($F6:$H6, N22:N24) into cell M23 and then copy cell M23 to the cell range F23:M31. This formula implements the backwards recursion for the explicit finite difference method. The cell F27 contains the option price at time step $i = 0$ for stock price step $j = 5$. For stock price of 20.00 the option price is 1.98, which is quite close to the analytic value of 2.01.

Next we show how to modify our explicit method for an European put option to price an **American put option**. We use same parameters as for the European put option.

For the American put option, the calculations for coefficients and option values at boundary conditions (8), (12) and (13) are the same as for the European one.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
17															
18															
19						i									
20	**analytic valuation**					0	1	2	3	4	5	6	7	8	
21				j	S	8	7	6	5	4	3	2	1	0	
22	d1	0.048540		0	0	**$20.31**	**$20.40**	**$20.48**	**$20.57**	**$20.65**	**$20.74**	**$20.83**	**$20.91**	**$21.00**	
23	d2	-0.182400		1	4	$16.31	$16.40	$16.48	$16.57	$16.65	$16.74	$16.83	$16.91	**$17.00**	
24	Nd1	0.519357		2	8	$12.31	$12.40	$12.48	$12.57	$12.65	$12.74	$12.83	$12.91	**$13.00**	
25	Nd2	0.427634		3	12	$8.34	$8.41	$8.49	$8.57	$8.66	$8.74	$8.83	$8.91	**$9.00**	
26	C	$1.70		4	16	$4.63	$4.65	$4.68	$4.71	$4.74	$4.79	$4.84	$4.91	**$5.00**	
27	**P**	**$2.01**		5	20	**$1.98**	$1.90	$1.82	$1.72	$1.62	$1.50	$1.36	$1.19	**$1.00**	
28				6	24	$0.71	$0.64	$0.56	$0.48	$0.40	$0.31	$0.21	$0.11	**$0.00**	
29				7	28	$0.23	$0.19	$0.15	$0.11	$0.07	$0.04	$0.02	$0.00	**$0.00**	
30				8	32	$0.07	$0.05	$0.03	$0.02	$0.01	$0.00	$0.00	$0.00	**$0.00**	
31				9	36	$0.02	$0.01	$0.01	$0.00	$0.00	$0.00	$0.00	$0.00	**$0.00**	
32				10	40	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	
33															

Figure 11: **Option prices for the explicit finite difference method for pricing a European put option**. At a current market price of the underlying asset of $20, and market and contractual conditions outlined in figure 10 above, the bolded cell on the left (B27) gives the theoretical value of the European put option price. The boxed and bolded cell to its right (F27) is our approximation to this European put option price.

This is shown in figure 12.

We put `Excel` code =MAX(MAX($B$5-$D23*$B$8,0),MMULT($F6:$H6,N22:N24)) in M23. This computes the value of the option for the time step $i = 7$ (one time unit before maturity) and for the stock price S=$4. This `Excel` code computes the higher of the early exercise price and the price computed using the explicit finite difference method.

We then copy the formula `Excel` code in M23 to F23:M31 to automatically compute the other option values. The American option value at time 0 for stock price $20.00 is $2.06 based on these calculations. This is higher than the value of the equivalent European option, as it should be.

We have now shown how to implement both the implicit and explicit finite difference methods using `Excel`. `Excel` may not be the most suitable software package for implementation of these methods in industrial practice. However, as many students and industry practitioners in the financial industry use `Excel` and understand it well, this implementation may aid their understanding of how the method works, and assist in the creation of software for implementing it more efficiently, as well as providing a check on the results.

				i								
				0	1	2	3	4	5	6	7	8
	j	S		8	7	6	5	4	3	2	1	0
	0	0		**$21.00**	**$21.00**	**$21.00**	**$21.00**	**$21.00**	**$21.00**	**$21.00**	**$21.00**	**$21.00**
	1	4		$17.00	$17.00	$17.00	$17.00	$17.00	$17.00	$17.00	$17.00	**$17.00**
	2	8		$13.00	$13.00	$13.00	$13.00	$13.00	$13.00	$13.00	$13.00	**$13.00**
	3	12		$9.00	$9.00	$9.00	$9.00	$9.00	$9.00	$9.00	$9.00	**$9.00**
	4	16		$5.00	$5.00	$5.00	$5.00	$5.00	$5.00	$5.00	$5.00	**$5.00**
	5	20		**$2.06**	$1.97	$1.88	$1.77	$1.65	$1.51	$1.36	$1.19	**$1.00**
	6	24		$0.73	$0.65	$0.57	$0.49	$0.40	$0.31	$0.21	$0.11	**$0.00**
	7	28		$0.23	$0.19	$0.15	$0.11	$0.07	$0.04	$0.02	$0.00	**$0.00**
	8	32		$0.07	$0.05	$0.03	$0.02	$0.01	$0.00	$0.00	$0.00	**$0.00**
	9	36		$0.02	$0.01	$0.01	$0.00	$0.00	$0.00	$0.00	$0.00	**$0.00**
	10	40		**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**	**$0.00**

Figure 12: **Option prices for the explicit finite difference method of pricing an American put option**. The cell formulæ of figure 11 above are altered to allow for the possibility of early exercise. Compared to the European put option price results in figure 11, we see the American put option prices are never smaller, and often larger.

4 Conclusion

We have explained above how we can convert the PDE for an option price together with its boundary condition into a difference equation by using finite difference approximations to the derivatives within the PDE. We have also shown how the difference equation can be solved and presented an `Excel` implementation of the solution to the difference equation. This paper is based on our classroom experience of teaching senior actuarial and finance students about the finite difference method and how it can be used to provide an approximate valuation of options. We were using the well known textbook Hull (2012) for the course we were teaching.

In our experience most finance and actuarial students find the exposition of this topic via an `Excel` implementation to be an easy way to learn both the underlying ideas and their implementation. We acknowledge that `Excel` is not the most appropriate software to use for an industrial application of the finite difference methods. However, most of our finance and actuarial students have `Excel` expertise but lack expertise in PDE theory and in other software tools. Using `Excel` to implement

the methods has pedagogical benefits compared to other software tools—students can see how the method works on the computer screen and the inputs, intermediate calculations and final results are all displayed to the user. This works well for small scale examples and makes the method much easier to understand. This enhances student learning of the material and eliminates blockages to learning due to the difficulty of implementation via pen, paper and calculator or via other, more opaque, software packages.

References

Andreasen, Jesper, Bjarke Jensen and Roy Poulsen. 1998. "Eight valuation methods in financial mathematics: the Black-Scholes formula as an example." *Mathematical Scientist* 23(1):18–40.

Black, Fischer and Myron Scholes. 1973. "The Pricing of Options and Corporate Liabilities." *The Journal of Political Economy* 81(3):637–654.

Boyle, Phelim P. 1986. "Option valuation using a three-jump process." *International Options Journal* 3(1):7–12.

Cox, John C., Stephen A. Ross and Mark Rubinstein. 1979. "Option pricing: A simplified approach." *Journal of Financial Economics* 7(3):229–263.

Duffy, Daniel J. 2006. *Finite Difference Methods in Financial Engineering : A Partial Differential Equation Approach*. John Wiley & Sons, Ltd.

Hackbusch, Wolfgang. 2014. *The Concept of Stability in Numerical Mathematics*. Springer Series in Computation Mathematics 45 Berlin: Springer-Verlag.

Hull, John. 2012. *Options, Futures and Other Derivatives*. 8th ed. Englewood Cliffs, NJ, USA: Prentice Hall.

John, Fritz. 1982. *Partial Differential Equations*. Applied Mathematical Sciences 1 4th ed. Springer-Verlag.

Lucchetti, Roberto. 2006. *Convexity and Well-Posed Problems*. CMS Books in Mathematics Springer-Verlag.

Press, William H., Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. 2007. *Numerical Recipes : The Art of Scientific Computing*. 3rd ed. Cambridge University Press.

Shapira, Yair. 2006. *Solving PDEs in C++ : Numerical Methods in a Unified Object-Oriented Approach*. SIAM.

Wilmott, Paul. 2013. *Paul Wilmott on Quantitative Finance*. John Wiley & Sons.

Wilmott, Paul, Jeff Dewynne and Sam Howison. 1993. *Option Pricing: Mathematical Models and Computation*. Oxford Financial Press.

Wilmott, Paul, Sam Howison and Jeff Dewynne. 1995. *The mathematics of financial derivatives: a student introduction*. Cambridge University Press.