Solving the Black Scholes Equation using a Finite Difference Method

Daniel Hackmann

12/02/2009

1 Introduction

This report presents a numerical solution to the famous Black-Scholes equation. Although a closed form solution for the price of European options is available, the prices of more complicated derivatives such as American options require numerical methods. Practitioners in the field of financial engineering often have no choice but to use numerical methods, especially when assumptions about constant volatility and interest rate are abandoned. [1]

This report will focus primarily on the solution to the equation for European options. It first presents a brief introduction to options followed by a derivation of the Black Scholes equation. Then it will introduce the finite difference method for solving partial differential equations, discuss the theory behind the approach, and illustrate the technique using a simple example. Finally, the Black-Scholes equation will be transformed into the heat equation and the boundary-value problems for a European call and put will be solved. A brief analysis of the accuracy of the approach will also be provided.

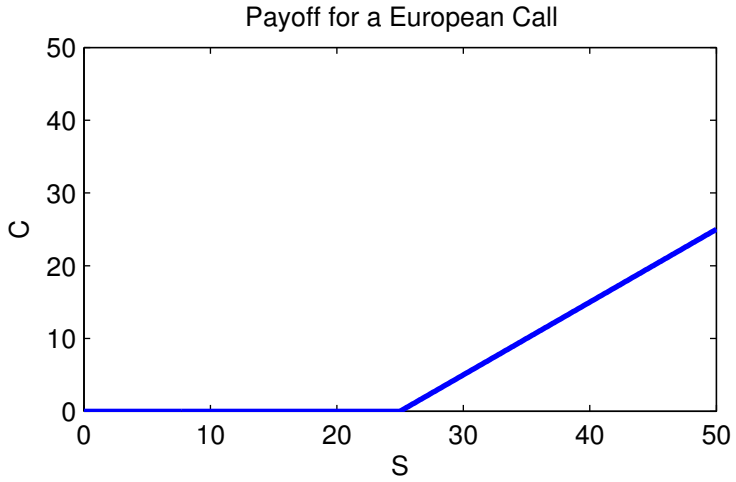
2 A brief introduction to options

A financial derivative is a financial product that derives its value from the value of another financial asset called the underlying asset. The derivatives of interest in this report are European option contracts. An European option contract is an agreement between two parties that gives one party the right to buy or sell the underlying asset at some specified date in the future for a fixed price. This right has a value, and the party that provides the option will demand compensation at the inception of the option. Further, once initiated, options can be traded on an exchange much like the underlying asset.

The options under consideration in this report fall into two classes: calls and puts. A European call option on a stock gives the buyer the right but not the obligation to buy a number of shares of stock (the underlying asset for stock options) for a specified price (E) at a specified date (T) in the future. If the price of the stock (S) is below the exercise price, the call option is said to be "out of the money" whereas a call option with an exercise price below the stock price is classified as "in the money". The value of the option is thus a function of S and time, and its payoff ,which describes the option's value at date T , is given by

$$C(S,T) = \max(S - E, 0) \tag{1}$$

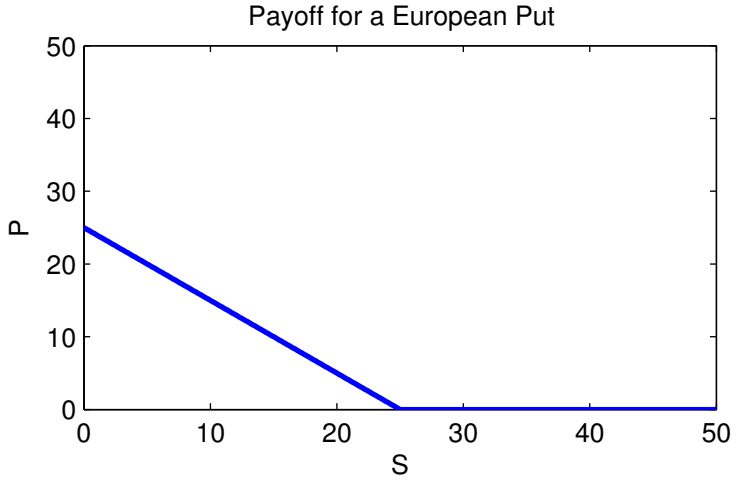
The graph below illustrates the payoff for a European call option with exercise price $E = 25$.



A European put option gives the buyer the right to sell a number of shares of stock for a specified price at a specified time. The payoff which describes a put option's value at time T is given by

$$P(S,T) = \max(E - S, 0) \tag{2}$$

The graph below illustrates the payoff for a European put option with exercise price $E = 25$.



European options are some of the simplest financial derivatives. They are interesting, because their valuation proved to be difficult until 1973. Before that, there was no generally accepted model that could give an options trader the

value of an option before expiry. The answer was provided by solving the Black-Scholes differential equation. The derivation of Fischer Black, Myron Scholes and Robert Merton's differential equation follows in the next section.

3 The Black Scholes analysis

In order to develop a model for the price of a stock option, it is necessary to first develop a model for the price of the stock itself. Economic theory and historical data suggest that stock returns are composed of two components. First, the value of a stock will increase with time at a rate μ known as the drift rate. Second, at any time the value of the stock is subject to random variability. This variability is expressed by a random variable Z with certain special properties.

The notion that a stock's value will necessarily increase over time is due to the expectation that a company will, in the long run, generate a return for investors. This return is measured as a percentage of the value of the investment, so that the expected absolute return is dependent on the price of the stock. The price change of a risk-less stock in a time interval Δt could thus be modeled as follows:

$$\Delta S = S\mu\Delta t \tag{3}$$

It is clear, however, that no stock is risk-less. Risk is modeled by the stochastic term Z which has the following two properties:

1. $\Delta Z = \epsilon\sqrt{\Delta t}$ where ϵ is a standard normal random variable, i.e. $\epsilon \sim N(0,1)$.
2. The value of ΔZ in time interval Δt is independent of $\triangle Z$ in any other time interval.

These two properties describe Z as a random variable that follows a Wiener process. Notice that for n time steps:

$$E[\sum_{i=1}^{n} \Delta Z_i] = 0 \tag{4}$$

$$Var(\sum_{i=1}^{n} \Delta Z_i) = n\Delta t \tag{5}$$

Thus Z is a random variable whose expected future value depends only on its present value, and whose variability increases as more time intervals are considered. The first property is consistent with an economic theory called the weak form of market efficiency that states that current stock prices reflect all information that can be obtained from the records of past stock prices. The second property shows that uncertainty about the value of Z increases as one

looks further into the future. This is intuitively appealing, as more events affecting the stock price are anticipated in longer time intervals. For a more detailed discussion of this concept see. [6]

The first property of Z implies that if Δt represents one unit of time, $Var(\Delta Z) = 1$. To reflect the risk of a particular stock, the variability of ΔZ needs to be adjusted. That is, it must be scaled by a quantity which is an estimate of the variability of S in Δt. This quantity is σS where σ is the standard deviation of S during Δt expressed as a percentage. Thus $\sigma S \Delta Z$ is a random variable with standard deviation σS. It is now possible to model the behaviour of a stock price as follows:

$$\Delta S = \mu S \Delta t + \sigma S \Delta Z \tag{6}$$

As $\Delta t \to 0$ this becomes:

$$dS = \mu S dt + \sigma S dZ \tag{7}$$

Formally, (7) implies that S follows an Ito process.

The Black-Scholes analysis assumes that stock prices behave as just demonstrated. Further assumptions of the analysis are as follows:

1. Investors are permitted to short sell stock. That is, it is permitted to borrow stock with the intent to sell and use the proceeds for other investment. Selling an asset without owning it is known as having a short position in that asset, while buying and holding an asset is known as having a long position.
2. There are no transaction costs or taxes.
3. The underlying stock does not pay dividends.
4. There are no arbitrage opportunities. That is, it is not possible to make risk free investments with a return greater than the risk free rate.
5. Stock can be purchased in any real quantity. That is, it is possible to buy π shares of stock or $1/100$ shares of stock.
6. The risk free rate of interest r is constant.

With these assumptions, one can begin to consider the value of a call or a put option. The following section will use $V(S,t)$ to denote the value of the option in order to emphasize that the analysis is independent of the form of the financial asset under consideration. As long as the value of the asset depends only on S and t this analysis holds. To understand the form that $V(S,t)$ will take it is necessary to use a result from stochastic calculus known as Ito's Lemma. This states that if x follows a general Ito process

$$dx = a(x,t)dt + b(x,t)dZ \tag{8}$$

and $f = f(x,t)$, then

$$df = \left(\frac{\partial f}{\partial x}a(x,t) + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(b(x,t))^2\right)dt + \frac{\partial f}{\partial x}b(x,t)dZ \tag{9}$$

Applying Ito's lemma to the function $V(S,t)$ yields

$$dV = \left(\frac{\partial V}{\partial S}\mu S + \frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right)dt + \frac{\partial V}{\partial S}\sigma S dZ \tag{10}$$

This rather complicated expression is difficult to solve for V , especially since it contains the stochastic term dZ . The revolutionary idea behind the Black Scholes analysis is that one can create a portfolio consisting of shares of stock and derivatives that is instantaneously risk-less, and thus eliminates the stochastic term in (10). The instantaneously risk-less portfolio at time t consists of one long position in the derivative and a short position of exactly $\frac{\partial V}{\partial S}$ shares of the the underlying stock. The value of this portfolio is given by

$$\Pi = V - \frac{\partial V}{\partial S}S \tag{11}$$

and the instantaneous change of Π is

$$d\Pi = dV - \frac{\partial V}{\partial S}dS \tag{12}$$

Combining (7), (10), and (12) one obtains

$$d\Pi = -\frac{\partial V}{\partial S}dS + \left(\frac{\partial V}{\partial S}\mu S + \frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right)dt + \frac{\partial V}{\partial S}\sigma S dZ$$

$$d\Pi = -\frac{dV}{dS}\left(\mu S dt + \sigma S dZ\right) + \left(\frac{\partial V}{\partial S}\mu S + \frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right)dt + \frac{\partial V}{\partial S}\sigma S dZ$$

$$d\Pi = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2\right)dt \tag{13}$$

As anticipated, the change in the instantaneously risk-less portfolio is not dependent on the stochastic term dZ . It should be noted that in order for the portfolio to maintain its risk-less property it must be rebalanced at every point in time as $\frac{\partial V}{\partial S}$ will not remain the same for different values of t . Thus shares will need to be bought and sold continuously in fractional amounts as was stipulated in the assumptions.

Since this portfolio is risk-less, the assumption that there are no arbitrage opportunities dictates that it must earn exactly the risk free rate. This implies that the right hand side of (13) is equal to $r\Pi dt$ where r represents the risk free rate of return. Setting the right hand side of (13) equal to $r\Pi dt$ and using (11) one obtains the famous Black-Scholes partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 + \frac{\partial V}{\partial S}rS - rV = 0 \tag{14}$$

As mentioned in the introduction, for both European calls and puts, analytic solutions for $V(S,t)$ have been found. In order to solve the problem for either option, one first needs to specify boundary and final conditions. These are obtained by making financial arguments about the behaviour of $C(S,t)$ and $P(S,t)$ at time T and for extreme values of S . Final conditions have already been introduced in section 2. At time T the option is either in or out of the money, and is thus either worthless or has a positive value. Also, should S ever reach a price of 0, it is clear from (7) that dS must equal 0, so that no further change in S is possible. This implies that the call option will not be exercised and $C(0,t) = 0$, while the put will surely be exercised and will have a value of $P(0,t) = Ee^{-r(T-t)}$. The term $e^{-r(T-t)}$ is added in this last expression to express the present value of the exercise price at time $T - t$. This is necessary, because $P(0,T)$ will surely be worth E , which is equivalent to $Ee^{-r(T-t)}$ invested at the risk free rate for a period of $T - t$. Finally, as $S \to \infty$ the call option will almost certainly be exercised, so that $C(S,t) \sim S - Ee^{-r(T-t)}$, while it becomes very unlikely that the put option will be exercised so that $P(S,t) \to 0$. Here $\sim$ is used in the context of order notation, so that $f(x) \sim g(x)\ as\ x \to a \iff \lim_{x\to a} \frac{f(x)}{g(x)} = 1$.[7]

To summarize, the boundary conditions for a European call are given by

$$C(S,T) = \max(S - E, 0)\ ,\ S > 0 \tag{15}$$

$$C(0,t) = 0\ ,\ t > 0 \tag{16}$$

$$C(S,t) \sim S - Ee^{-r(T-t)}\ as\ S \to \infty\ ,\ t > 0 \tag{17}$$

While the boundary conditions for the European put are given by

$$P(S,T) = \max(E - S, 0)\ ,\ S > 0 \tag{18}$$

$$P(0,t) = Ee^{-r(T-t)}\ ,\ t > 0 \tag{19}$$

$$P(S,t) \to 0\ as\ S \to \infty\ ,\ t > 0 \tag{20}$$

Two features of these problems make them difficult to solve directly. The first is that instead of posing an initial condition with respect to time, the problems present a final condition given by equations (15) and (18). Secondly, the boundary conditions given by (17) and (18) depend on t .

4 Finite difference methods

Finite difference methods involve approximations of the derivative at points of interest in the domain of the solution of a differential equation. In simple cases, the derivative of a function of one variable can be approximate near a point x by using the definition of the derivative

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

because when h is small one expects that

$$\frac{df(x)}{dx} \approx \frac{f(x+h) - f(x)}{h} \tag{21}$$

In equations with more than one variable, and for derivatives of higher order, derivatives can instead be approximated by a Taylor series expansion. In particular, if one was interested in obtaining an approximation for the first derivative of f with respect to x near the point (x, y) one would proceed as follows

$$f(x+h, y) = f(x, y) + \frac{\partial f(x, y)}{\partial x} \cdot h + O(h^2)$$

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x+h, y) - f(x, y)}{h} \tag{22}$$

where $O(h)$ represents the order of the remaining terms in the series. The approximation of the derivative given in (22) is called the forward difference because it approximates $\dfrac{\partial f}{\partial x}$ when a small quantity, h , is added to the x coordinate of the point (x, y) . One can also obtain the backward difference by considering the Taylor series expansions for $f(x-h, y)$ or the central difference by combining the results of the forward and backward differences. To compute approximations for second derivatives, one has the Taylor expansions:

$$f(x+h, y) = f(x, y) + \frac{\partial f(x, y)}{\partial x} h + \frac{1}{2} \frac{\partial^2 f(x, y)}{\partial x^2} h^2 + \frac{1}{6} \frac{\partial^3 f(x, y)}{\partial x^3} h^3 + O(h^4) \tag{23}$$

$$f(x-h, y) = f(x, y) - \frac{\partial f(x, y)}{\partial x} h + \frac{1}{2} \frac{\partial^2 f(x, y)}{\partial x^2} h^2 - \frac{1}{6} \frac{\partial^3 f(x, y)}{\partial x^3} h^3 + O(h^4) \tag{24}$$

Upon adding (23) and (24) one obtains

$$\frac{\partial^2 f(x,y)}{\partial x^2} \approx \frac{f(x+h,y) - 2f(x,y) + f(x-h,y)}{h^2} \tag{25}$$

With these results, it is now possible to consider a basic example to demonstrate how this method works.

Example: The heat equation with finite boundary conditions

The heat equation in one spatial variable x is a partial differential equation which models the distribution of heat in a thin wire over time t. Consider the following problem

$$\frac{\partial U}{\partial t} = k^2 \frac{\partial^2 U}{\partial x^2} \tag{26}$$

$$0 < x < 1, \ t > 0$$

$$U(x,0) = \sin \pi x$$

$$U(0,t) = U(1,t) = 0$$

where $\frac{\partial U}{\partial t}$ and $\frac{\partial^2 U}{\partial x^2}$ are the first and second partial derivatives of the temperature $U(x,t)$ with respect to time and space respectively and k is a constant.[1] To begin the finite difference approximation one must choose a rectangular region in the domain of $U(x,t)$ and partition it into points of interest to form a grid. The length of the rectangle along the x axis is determined by the boundary conditions. For this example one would choose the interval $[0,1]$. For t the choice must consider the initial condition, so the length in t will be the interval $[0,T]$. Let $T = 0.16$ and divide the intervals for x and t into $M = 5$ and $N = 2$ equally spaced increments of $\Delta x = 0.2$ and $\Delta t = 0.08$ respectively. The goal is to approximate the value of the function U at the $(M-1) \times (N) = 8$ points of interest in the domain $[0,1] \times [0,0.16]$. These points are referred to as the finite difference mesh and have coordinates (x_i, t_j) where i and j are indices for the time and space increments. With the approximation of the derivative obtained in (22) and (25) one can estimate (26) by

$$\frac{U(x,t+\Delta t) - U(x,t)}{\Delta t} = k^2 \frac{U(x+\Delta x,t) - 2U(x,t) + U(x-\Delta x,t)}{(\Delta x)^2} \tag{27}$$

[1] Example taken from [3] but solved independently using code in Appendix A

To simplify (27), one can label the unknown quantities $U(x_i, t_j)$ with u_i^j for $i = 1, 2, 3, 4$ and $j = 1, 2$ and write

$$u_i^{j+1} = \frac{k^2 \Delta t}{(\Delta x)^2} \left(u_{i+1}^j - 2u_i^j + u_{i-1}^j \right) + u_i^j \tag{28}$$

The equality in (28) is known as the explicit method of approximating the solution to (26). The term explicit is used to describe the nature of the relationship of the u_i^j with respect to time. Specifically, one can express the term u_i^{j+1} explicitly in terms of the approximations of U at time t_j , the time immediately prior to t_{j+1} . Although the explicit method is the easiest to implement computationally, convergence to a solution depends on the ratio of Δt and Δx^2 . Note that the forward difference was used in (27) to approximate $\dfrac{\partial U}{\partial t}$. Had the backward difference been used instead, (28) would have the fully-implicit form

$$u_i^j = u_i^{j+1} - \frac{k^2 \Delta t}{(\Delta x)^2} \left(u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1} \right) \tag{29}$$

This method is more difficult to implement computationally, but does not depend on the ratio of Δt and Δx^2 for convergence. Note that with this approach approximations of U at time t_{j+1} cannot be explicitly expressed in terms of approximations of U at time t_j . A third approach that converges independently of the ratio of Δt and Δx^2 and faster than both (28) and (29), relies on taking a weighted average of the expressions for (28) and (29). This yields

$$u_i^{j+1} = \Delta t k^2 \left[\alpha \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2} + (1 - \alpha) \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{(\Delta x)^2} \right] + u_i^j \tag{30}$$

When $\alpha = \dfrac{1}{2}$ (30) becomes the Crank-Nicolson approximation. Returning to the example (26), for $k^2 = 1$ this yields

$$3u_i^{j+1} = -u_i^j + u_{i+1}^j + u_{i-1}^j + u_{i+1}^{j+1} + u_{i-1}^{j+1} \tag{31}$$

which represents a linear system of 8 equations in 8 unknowns. Solving this system requires constructing a matrix and solving the following equation for the u_i^j

$$\begin{bmatrix} 3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & -1 & 3 \\ 1 & -1 & 0 & 0 & 3 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 3 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_3^1 \\ u_4^1 \\ u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{bmatrix} =$$

$$\begin{bmatrix} -\sin(0.2\pi) + \sin(0.4\pi) + \sin(0) \\ 0 \\ -\sin(0.6\pi) + \sin(0.8\pi) + \sin(0.4\pi) \\ 0 \\ 0 \\ -\sin(0.4\pi) + \sin(0.6\pi) + \sin(0.2\pi) \\ 0 \\ -\sin(0.8\pi) + \sin(\pi) + \sin(0.6\pi) \end{bmatrix}$$

When compared with the known solution for the problem, $U(x,t) = e^{\pi^2 t}\sin(\pi x)$, which can be obtained using a separation of variables technique, the approximation is valid up to 2 decimal places for most values of x. The results comparing the numerical approximation to the exact solution are presented below. Considering that these approximations are generally conducted with a far greater number of steps and smaller increments for both x and t these results are encouraging.

x_i	u_i^1	$U(x_i, 0.08)$	Diff.
0.2	0.26287	0.26688	0.004012
0.4	0.42533	0.43182	0.006493
0.6	0.42533	0.43182	0.006493
0.8	0.26287	0.26688	0.004012

x_i	u_i^2	$U(x_i, 0.16)$	Diff.
0.2	0.11756	0.12117	0.003617
0.4	0.19021	0.19606	0.005852
0.6	0.19021	0.19606	0.005852
0.8	0.11756	0.12117	0.003617

The following section will demonstrate the theoretical validity of the Crank-Nicholson method.

5 Theoretical basis for the finite difference approach

The focus of this section is to demonstrate the validity of the methods used to obtain the solutions found in section 6. Since these results were achieved using the Crank-Nicolson approximation, this will be the primary method of interest. Additionally, it will be shown why the Crank-Nicolson approach is sometimes preferred to the simpler explicit scheme.

To verify that finite difference approximations are an accurate method of estimating the solution to the heat equation, it is necessary to establish that they are convergent. An approximation is said to be convergent if its values approach those of the analytic solution as Δx and Δt both approach zero.[5] Since convergence is difficult to prove directly, it is helpful to use an equivalent result known as the Lax Equivalence Theorem. It states that a consistent finite difference method used to solve a well-posed linear initial value problem is convergent if and only if it is stable. This introduces two new terms: consistent and stable.

1. Consistent: A method is consistent if its local truncation error $T_i^j \to 0$ as $\Delta x \to 0$ and $\Delta t \to 0$. Local truncation error is the error that occurs when the exact solution $U(x_i, t_j)$ is plugged into the difference approximation at each point of interest. For example, in (30) with $\alpha = \frac{1}{2}$

$$\begin{aligned} T_i^j(\Delta x, \Delta t) = U(x_i, t_{j+1}) \\ - \frac{k^2 \triangle t}{2} \left[\frac{U(x_{i+1}, t_j) - 2U(x_i, t_j) + U(x_{i-1}, t_j)}{(\triangle x)^2} \right. \\ \left. + \frac{U(x_{i+1}, t_{j+1}) - 2U(x_i, t_{j+1}) + U(x_{i-1}, t_{j+1})}{(\triangle x)^2} \right] - U(x_i, t_j) \end{aligned} \quad (32)$$

2. Stable: A method is stable if errors incurred at one stage of the approximation do not cause increasingly larger errors as the calculation continues.

It is known that the Crank-Nicolson method is both consistent and stable in approximating solutions to the heat equation. To demonstrate consistency, one must proceed as in (32) and enter the exact solution into the difference equation for the Crank-Nicolson scheme. Here the exact solution is given by the full Taylor series expansion of U.

$$\begin{aligned} T_i^j(\Delta x, \Delta t) = U(x_i, t_j) + \frac{\partial U(x_i, t_j)}{\partial t} \Delta t + O((\triangle t)^2) \\ - \frac{k^2 \triangle t}{2 (\triangle x)^2} \left[U(x_i, t_j) + \frac{\partial U(x_i, t_j)}{\partial x} \Delta x + \frac{\partial^2 U(x_i, t_j)}{2 \partial x^2} (\Delta x)^2 + O((\triangle x)^3) - 2U(x_i, t_j) \right. \\ + U(x_i, t_j) - \frac{\partial U(x_i, t_j)}{\partial x} \Delta x + \frac{\partial^2 U(x_i, t_j)}{2 \partial x^2} (\Delta x)^2 + O((\triangle x)^3) \\ + U(x_i, t_{j+1}) + \frac{\partial U(x_i, t_{j+1})}{\partial x} \Delta x + \frac{\partial^2 U(x_i, t_{j+1})}{2 \partial x^2} (\Delta x)^2 + O((\triangle x)^3) \\ - 2U(x_i, t_{j+1}) + U(x_i, t_{j+1}) - \frac{\partial U(x_i, t_{j+1})}{\partial x} \Delta x \\ \left. + \frac{\partial^2 U(x_i, t_{j+1})}{2 \partial x^2} (\Delta x)^2 + O((\triangle x)^3) \right] - U(x_i, t_j) \end{aligned} \quad (33)$$

Upon simplification (using the fact $\frac{\partial^2 U}{\partial x^2} = \frac{\partial U}{\partial t}$) one obtains

$$T_i^j(\Delta x, \Delta t) = \left(\frac{\partial U(x_i, t_j)}{\partial t}(1 - \frac{k}{2}^2) - \frac{k^2}{2}\frac{\partial U(x_i, t_{j+1})}{\partial t} \right) \Delta t + O((\triangle t)^2) + O((\triangle t(\triangle x)^2) \quad (34)$$

As (34) shows $T_i^j \to 0$ as $\Delta x \to 0$ and $\Delta t \to 0$ which demonstrates that the Crank-Nicholson method is indeed consistent.

To demonstrate stability, it is easier to first consider the explicit method and then apply a similar result to the Crank-Nicolson method.[2] Notice that in the case of the explicit method, one can write the difference equations in (28) as a matrix equation:

$$\begin{bmatrix} u_1^{j+1} \\ u_2^{j+1} \\ . \\ . \\ . \\ u_{M-1}^{j+1} \end{bmatrix} = \begin{bmatrix} (1-2r) & r & & & & \\ r & (1-2r) & r & & & \\ & & . & & & \\ & & & . & & \\ & & & & . & \\ & & & & r & (1-2r) \end{bmatrix} \begin{bmatrix} u_1^j \\ u_2^j \\ . \\ . \\ . \\ u_{M-1}^j \end{bmatrix} \quad (35)$$

In (35) $r = \frac{k^2 \Delta t}{(\Delta x)^2}$ and it is assumed, for simplicity, that a change of variable has been introduced to give boundary conditions $u_0^j = u_M^j = 0$. This is always possible for fixed end boundary conditions. Let u^{j+1} be the vector on the left hand side of (35), u^j the vector on the right hand side, and A the coefficient matrix. Also note that u^0 , the vector at t_0, is given by the initial condition, so that one can determine u at later times by solving (35) successively

$$u^1 = Au^0$$

$$u^2 = Au^1 = A^2u^0$$

$$...$$

$$u^N = Au^{N-2} = A^2u^{N-3} = ... = A^Nu^0 \quad (36)$$

In (36) superscripts on the u terms denote time increments while they denote exponents on A . It can be demonstrated, that if an error is introduced at any point in this process, then the error propagates via the same iterative calculation. For example, the error e^j at time t_j that is attributable to the error e^0 at time t_0 is given by

$$e^j = A^j e^0 \quad (37)$$

[2]Proof abbreviated from [5]

Further, it can be shown that A has distinct eigenvalues $\lambda_1, \lambda_2, ..., \lambda_{M-1}$ and thus has linearly independent eigenvectors $x_1, x_2, ..., x_{M-1}$. For constants, $c_1, c_2, ..., c_{M-1}$, it is then possible to write e^0 as a linear combination of eigenvectors

$$e^0 = c_1 x_1 + c_2 x_2 + ... + c_{M-1} x_{M-1} \tag{38}$$

Subsequently (37) becomes

$$e^j = A^j e^0 = \sum_{i=1}^{M-1} A^j c_i x_i = \sum_{i=1}^{M-1} c_i A^j x_i = \sum_{i=1}^{M-1} c_i \lambda_i^j x_i \tag{39}$$

This demonstrates that errors will not increase with each iteration if $|\lambda_i| \leq 1$ for all i . That is, the explicit method is stable if the magnitude of the largest eigenvalue of the coefficient matrix is smaller than one. The distinct eigenvalues of A are given by

$$1 - 4r \sin^2 \left(\frac{m\pi}{2M} \right) \ , \ m = 1, 2, ..., M-1$$

For stability, the following condition must hold

$$-1 \leq 1 - 4r \sin^2 \left(\frac{m\pi}{2M} \right) \leq 1$$

This leads to the conclusion that $r \leq \frac{1}{2}$ in order for the explicit scheme to be stable because the preceding implies that

$$r \leq \frac{1}{2} \left(\frac{1}{\sin^2 \left(\frac{m\pi}{2M} \right)} \right)$$

Since $r = \dfrac{k^2 \Delta t}{(\Delta x)^2}$ this demonstrates that stability and hence convergence of the explicit method depends on the choice of Δt and Δx . For example, if one wanted to double the space increments of a stable solution, one would be required to quadruple the time increments in order to maintain stability. This can lead to unnecessarily large and computationally intensive calculations.

The Crank-Nicolson scheme, by comparison, does not require restrictions on r for stability. Writing (30) in matrix form with $\alpha = \frac{1}{2}$ and under the assumption

that the boundary conditions are zero yields

$$\begin{bmatrix} (2+2r) & r & & & & \\ r & (2+2r) & r & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & r & (2+2r) \end{bmatrix} \begin{bmatrix} u_1^{j+1} \\ u_2^{j+1} \\ \cdot \\ \cdot \\ \cdot \\ u_{M-1}^{j+1} \end{bmatrix} =$$

$$\begin{bmatrix} (2-2r) & r & & & & \\ r & (2-2r) & r & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & r & (2-2r) \end{bmatrix} \begin{bmatrix} u_1^{j} \\ u_2^{j} \\ \cdot \\ \cdot \\ \cdot \\ u_{M-1}^{j} \end{bmatrix} \tag{40}$$

This results in the matrix equation

$$Bu^{j+1} = Au^j \tag{41}$$

If one performs the same analysis as for the explicit method with the coefficient matrix $B^{-1}A$ one finds, as previously, that stability depends on the magnitude of the eigenvalues. For $B^{-1}A$ these are given by

$$\frac{2 - 4r \sin\left(\frac{m\pi}{2(M-2)}\right)}{2 + 4r \sin\left(\frac{m\pi}{2(M-2)}\right)}, \ m = 1, 2, ..., M-1 \tag{42}$$

From (42) it is clear that the magnitude of the eigenvalues is less than one regardless of the value of r. As a result the Crank-Nicolson method is stable without restrictions on the size of Δt and Δx . The preceding analysis has shown that the Crank-Nicolson is a convergent and convenient method of approximating the solution of the heat equation. Although it is not immediately obvious, this implies that the method is also useful in finding a numerical solution to the Black-Scholes equation.

6 Solution of the Black-Scholes equation

The previous section has demonstrated that finite difference methods, the Crank-Nicholson method in particular, are valid approaches for approximating solutions to boundary value problems for the heat equation. As mentioned in the introduction, the Black-Scholes equation can be transformed into the heat equation, which is how closed form solutions are obtained. This also means that the results of section 5 can be applied to solve the transformed equation. Both (14) and (26) are classified as parabolic linear partial differential equations (PDE's) of second order. They differ in that the heat equation is forward parabolic while (14) is backward parabolic. The distinction arises from the direction of the sign of the second partial derivative with respect to space (S or x) and the sign of the

derivative with respect to time. In forward parabolic equations the sign of the two derivatives is opposite when both terms are on one side of the equality, while in backward parabolic equations the sign is the same. For the purposes of this analysis it suffices to say that backward parabolic differential equations require final conditions in order to guarantee unique solutions, and forward differential equations require initial conditions.

The closed form solution of the Black-Scholes PDE is obtained by transforming (14) into the heat equation on an infinite interval.[3] To do so, one must employ a change of variables

$$S = Ee^x \ , \ t = T - \tau / \left(\tfrac{1}{2}\sigma\right) \ , \ C = Ev(x,\tau) \tag{43}$$

When these are substituted into (14) one gets a forward parabolic equation with one remaining dimensionless parameter $k = r/\left(\frac{1}{2}\sigma\right)$

$$\frac{\partial v}{\partial \tau} = \frac{\partial^2 v}{\partial x^2} + (k-1)\frac{\partial v}{\partial x} - kv$$

One further change of variable

$$v = e^{-\frac{1}{2}(k-1)x - \frac{1}{4}(k+1)^2\tau} u(x,\tau) \tag{44}$$

then gives the heat equation one an infinite interval.

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} \ , \ -\infty < x < \infty \ , \ \tau > 0 \tag{45}$$

This change of variables gives initial conditions for the call (15) and the put (18) respectively

$$u(x,0) = \max(e^{\frac{1}{2}(k+1)x} - e^{\frac{1}{2}(k-1)x}, 0) \tag{46}$$

$$u(x,0) = \max(e^{\frac{1}{2}(k-1)x} - e^{\frac{1}{2}(k+1)x}, 0) \tag{47}$$

It is worth noting that the boundary conditions no longer depend on time. In fact, fixed boundary conditions no longer apply at all. Instead the behaviour of u as x approaches $\pm\infty$ becomes important. It can be shown that if u does not grow too fast, a closed form solution to the transformed equation of a European call can be found. By reversing the transformations, one finds the formula for the value of a call as presented in the 1973 paper by Black and Scholes [2], to be

$$C(S,t) = SN(d_1) - Ee^{-r(T-t)}N(d_2) \tag{48}$$

where

$$d_1 = \frac{\log\left(S/E\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\left(T-t\right)}$$

$$d_2 = \frac{\log\left(S/E\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\left(T-t\right)}$$

[3] Full transformation found in [7]

and N is cumulative normal density function. To obtain the formula for a put, the standard approach is to use a formula that relates the value of a put and a call, known as the put-call parity. This states

$$C - P = S - Ee^{-r(T-t)}$$

From this one obtains the value of the European put

$$P(S,t) = Ee^{-r(T-t)}N(-d_2) - SN(-d_1) \tag{49}$$

To use the Crank-Nicholson method to solve the Black-Scholes equation for a put or a call one can proceed as in the example in section 4. The only complication is that there are now no given boundaries for u in the x domain. That is, one must take an infinite number of Δx increments to solve the problem on an infinite interval. Since this is not possible, one instead takes suitably large x^+ and small x^- values to approximate a boundary for the solution in x. A rule of thumb for the size of x^+ and x^- is as follows

$$x^- = \min(x_0, 0) - \log(4)$$

$$x^+ = \max(x_0, 0) + \log(4) \tag{50}$$

Here, 0 corresponds to the transformed strike price E and x_0 to the transformed stock price for which the value of the option is being calculated. One then makes the approximation that u behaves at these boundaries as it would as $x \to \pm\infty$ which corresponds to $S \to 0$ and $S \to \infty$ in the financial variables. Using (16) and (17) and the change of variables (43) and (44) one has the following transformed boundaries for the call

$$u(x^-, \tau) = 0 \tag{51}$$

$$u(x^+, \tau) = e^{\frac{1}{2}(k+1)x + \frac{1}{4}(k+1)^2\tau} - e^{\frac{1}{2}(k-1)x + \frac{1}{4}(k-1)^2\tau} \tag{52}$$

Using (19) and (20) with the same change of variables yields the transformed put boundaries

$$u(x^-, \tau) = e^{\frac{1}{2}(k-1)x + \frac{1}{4}(k-1)^2\tau} \tag{53}$$

$$u(x^+, \tau) = 0 \tag{54}$$

One can then proceed exactly as if this were the example in section 4, with one additional step which is to transform the approximated values of u into terms of V with the following transformation

$$V = E^{\frac{1}{2}(1+k)}S^{\frac{1}{2}(1-k)}e^{\frac{1}{8}(k+1)^2\sigma^2(T-t)}u(\log(S/E), \tfrac{1}{2}\sigma^2(T-t))$$

The results for the solution of the call and the put are displayed below. [4] The number of increments of tau is 200 with $\triangle\tau = 0.000225$ while the number of space increments is equal to the size of the x interval determined by (50) divided by $\triangle x = 0.0225$. The range for the number of space increments is between 130 and 160. Selected results are presented for both the "in the money" and "out of the money" case for both the put and the call and for three different dates to maturity. In both cases, stock prices were chosen that were not too far "out of the money" because the comparison becomes somewhat meaningless when the calculated values are worthless in financial terms. The financial parameters are:

$$E = 10\ ,\ \sigma = 0.3\ ,\ r = 0.04$$

Results for a Call

Time Rem.	C-N at $S = 5$	Exact at $S = 5$	Diff.	% Diff.
3 months	8.68E-07	5.59E-07	-3.08E-07	-55.1290
6 months	0.00032	0.00030	-0.00002	-6.2327
1 year	0.01083	0.01074	-0.00009	-0.8313

Time Rem.	C-N at $S = 15$	Exact at $S = 15$	Diff.	% Diff.
3 months	5.1011	5.1010	-6.72E-05	-0.0013
6 months	5.2195	5.2194	-4.64E-05	-0.0001
1 year	5.5003	5.5005	0.00011	0.0021

Results for a Put

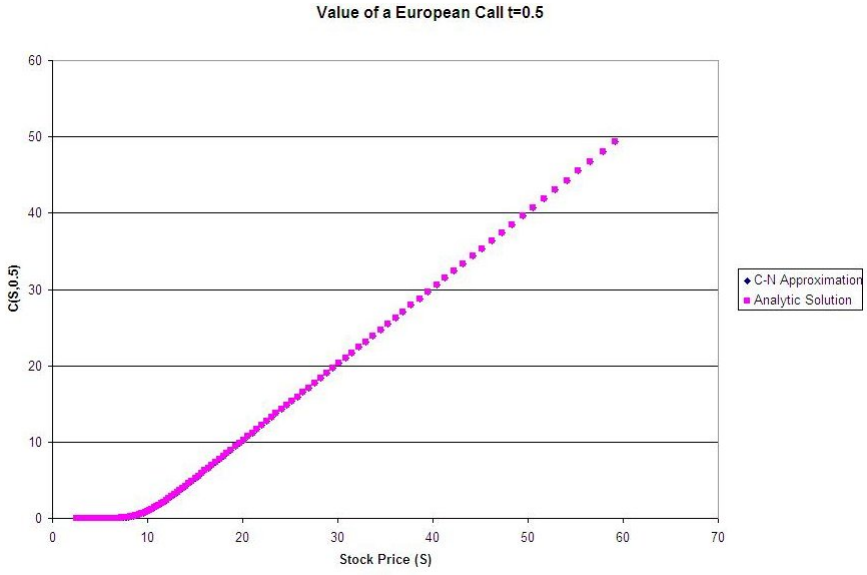
Time Rem.	C-N at $S = 7.5$	Exact at $S = 7.5$	Diff.	% Diff.
3 months	2.4169	2.4167	-0.00026	-0.0109
6 months	2.3916	2.3914	-0.00018	-0.0075
1 year	2.3985	2.3985	-0.00005	-0.0019

Time Rem.	C-N at $S = 12.5$	Exact at $S = 12.5$	Diff.	% Diff.

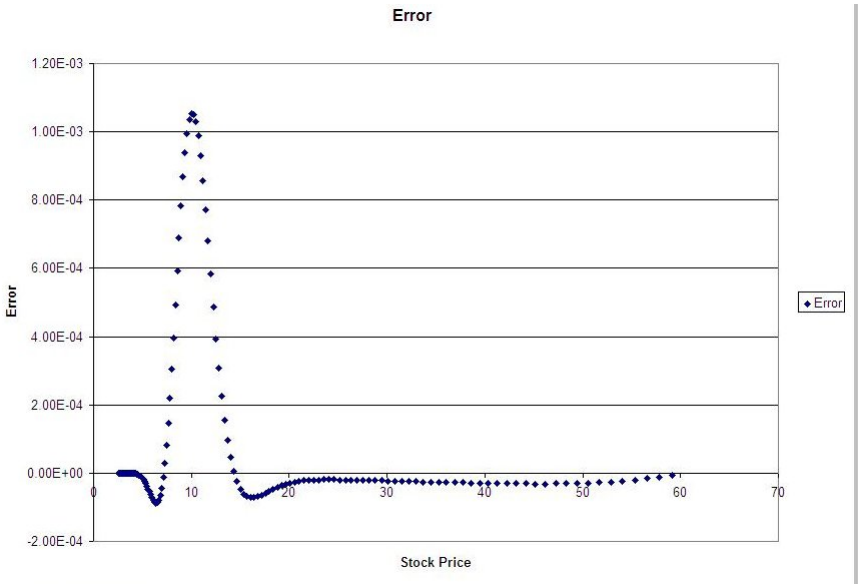
[4] Matlab code for a call and a put are displayed in Appendix B and C respectively. This method is not the most efficient way to solve the problem. Specifically, for the call and the put, it builds approximately 30,000 x 30,000 square matrix and solves the matrix equation as in the example of section 4 by using Matlab's "\" command for inverting the matrix. To build such a large matrix, it is necessary to use the "sparse" command because most personal computers will not store a matrix of this magnitude in memory. The "sparse" command tells Matlab to compress the matrix by storing only the non-zero terms. For a matrix that consists mainly of zeros, this greatly reduces the required memory although it can increase execution time for matrix operations. A better approach to solving this problem would likely have been to use the iterative matrix procedure in (40) which requires inverting 200 150 x 150 tri-diagonal matrices. At each step a more efficient method than matrix inversion like the LU factorization or a specific algorithm for tri-diagonal matrices could have been employed. This would have made for a more efficient process with faster execution time.

Time Rem.	C-N at $S = 12.5$	Exact at $S = 12.5$	Diff.	% Diff.
3 months	0.04307	0.04307	1.62E-06	0.0038
6 months	0.14615	0.14640	0.00025	0.1730
1 year	0.34156	0.34190	0.00034	0.1002

This demonstrates that the approximation is always accurate to three decimal places. Naturally it could be improved by increasing the number and decreasing the size of the space and time increments. A graph of all of the approximated values for the call with half a year remaining to maturity is shown below. These values were calculated under the assumption that $S = 15$ which is where the approximated solution should be least affected by the errors introduced by the approximated boundary conditions. Despite this, the graph of the actual and approximate solution are still close at large and small values of S indicating that the approximations at the boundary do not introduce large errors.



The next graph depicts the errors between the Crank-Nicolson approximation and the exact solution calculated for the same values of S and t . It demonstrates that the Crank-Nicolson solution oscillates around the exact solution when S is near the exercise price, and that the error is greatest in this vicinity as well. This is a well-known issue with the Crank-Nicolson approximation which is caused by the non-smooth final condition. [4] In practice, this problem is often overcome by introducing one or two smoothing steps using another approximation method, and beginning with the Crank-Nicolson approximation from there.



7 Conclusion

This report has presented an introduction to two related topics: the Black-Scholes partial differential equation; and the method of finite differences. It has demonstrated that the finite difference method is a useful, convergent, and relatively simple way to approximate solutions to the heat equation. In particular, the Crank-Nicolson method was successfully applied to the transformed boundary value problems of the European call and put options with results that were accurate to 3 decimal places. Finally, a brief analysis showed that errors were most pronounced, even at times before expiry, around the strike price where the final condition is not smooth.

A Appendix: Matlab code heat equation example

```
echo on
format short

% steps of x and t, m and n respectively
m=4;
n=2;

% maximum values of indices considering +1
M=m+1;
N=n;

% Upper and lower boundaries of x and t
% respectively and range of x and t respectively
bx1=0;
bx2=1;
bt1=0;
bt2=0.16;
rx=bx2-bx1;
rt=bt2-bt1;

% steps of x and t and values of x and t for each step
ix=rx/M;
it=rt/N;
x1=bx1:ix:bx2;
t1=bt1:it:bt2;

% heat equation constants
w=1;
y=(it*w^2)/(ix^2);
z=y/2;

% dimensions for matrix
k=m*n;

% gcd and lcm
g1 = gcd(m,n);
l1 = lcm(m,n);

% index beginning
j=0;
i=1;
l=1;
f=0;

% make matrix and vector for the equations
Q = zeros(k,1);
```

```
A = zeros(k,k);

% populate matrix
while l <= k
        if mod(l,l1) == 1 & l>l1
                j = 0 + f;
        elseif mod(l,l1) ~= 1 & j == N
                j = 0;
        end;
        if l <= k & i == M
                i = 1;
        end;
        J = [j+1,i,(1+y);j,i,(y-1);...
        j+1,i+1,-z;j+1,i-1,-z;j,i+1,-z;j,i-1,-z];
        for p = 1:6
                if J(p,1) == 0
                        Q(l,1) = Q(l,1) +...
                         sin(x1(J(p,2)+1)*pi)*(-1*J(p,3));
                elseif J(p,1) ~= 0 &...
                         J(p,2) ~= 0 & J(p,2) ~= M
                         A(l,(J(p,1)-1)*m + J(p,2)) = J(p,3);
                elseif J(p,1) ~= 0 & J(p,2) == 0...
                         | J(p,1) ~= 0 & J(p,2) == M
                         Q(l,1) = Q(l,1) + 0;
                end;
        end;
        j=j+1;
        i=i+1;
        if gcd(m,n) > 1 & mod(l,l1) == 0
                f=f+1;
        end;
        l=l+1;
end;

% solve and display results vector
S = zeros(k,5);
l=1;
i=int32(1);
j=int32(1);
for j = 1:n
        for i = 1:m;
                S(l,1) = j;
                S(l,2) = i;
                S(l,4) = exp(-(pi^2)*t1(j+1))...
                *sin(x1(i+1)*pi)
                l=l+1;
        end;
end;
S(:,3) = A\Q;
for l = 1:k
```

```
        S(l,5) = S(l,4) - S(l,3)
end;
disp(S);

echo off
```

B Appendix: Matlab code European call solution

```
echo on
format short

% steps of x and t, m and n respectively
m=140;
n=200;

% maximum values of indices considering +1
M=m+1;
N=n;

% Upper and lower boundaries of x and t
% respectively and range of x and t respectively
bx1=log(15/10)-(80*0.0225);
bx2=log(15/10)+(62*0.0225);
bt1=0;
bt2=0.045;
rx=bx2-bx1;
rt=bt2-bt1;

% steps of x and t and values of x and t for each step
ix=rx/M;
it=rt/N;
x1=bx1:ix:bx2;
t1=bt1:it:bt2;

%Black Scholes Vars
rr=0.04;
E=10;
sigma=0.3;
ak=rr/((sigma^2)/2);
S1 = E*exp(x1);
T1 = (t1/((sigma^2)/2));

% heat equation constants
w=1;
y=(it*w^2)/(ix^2);
z=y/2;

% dimensions for matrix
k=m*n;

% gcd and lcm
g1 = gcd(m,n);
l1 = lcm(m,n);
```

```
% index beginning
j=0;
i=1;
l=1;
f=0;

% make matrix and vector for the equations
Q = zeros(k,1);
A = sparse(k,k);

% populate matrix
while l <= k
        if mod(l,l1) == 1 & l>l1
                j = 0 + f;
        elseif mod(l,l1) ~= 1 & j == N
                j = 0;
        end;
        if l <= k & i == M
                i = 1;
        end;
        J = [j+1,i,(1+y);j,i,(y-1);j+1,i+1,-z;j+1,i-1,-z;...
        j,i+1,-z;j,i-1,-z];
        for p = 1:6
                if J(p,1) == 0
                        Q(l,1) = Q(l,1) + max((exp(0.5*(ak+1)...
                        *x1(J(p,2)+1))-exp(0.5...
                        *(ak-1)*x1(J(p,2)+1))),0)*(-1*J(p,3));
                elseif J(p,1) ~= 0 & J(p,2) ~= 0 & J(p,2) ~= M
                        A(l,(J(p,1)-1)*m + J(p,2)) = J(p,3);
                elseif J(p,1) ~= 0 & J(p,2) == 0
                        Q(l,1) = Q(l,1) + 0;
                elseif J(p,1) ~= 0 & J(p,2) == M
                        Q(l,1) = Q(l,1) + ((exp(x1(J(p,2)+1))...
                        -exp(-ak*t1(J(p,1)+1)))*(exp((0.5*(ak-1)...
                        *x1(J(p,2)+1))+(0.25*((ak+1)^2)...
                        *t1(J(p,1)+1)))))*(-1*J(p,3));
                end;
        end;
        j=j+1;
        i=i+1;
        if gcd(m,n) > 1 & mod(l,l1) == 0
                f=f+1;
        end;
        l=l+1;
end;

% solve and display results vector
S = zeros(k,8);
S(:,5) = (A\Q);
l=1;
```

```
i=int32(1);
j=int32(1);
for j = 1:n
        for i = 1:m;
                S(l,1) = j;
                S(l,2) = i;
                S(l,3) = T1(j+1);
                S(l,4) = S1(i+1);
                S(l,6) = S(l,5)*E*exp((-0.5*(ak-1)*x1(i+1))...
                +(-0.25*((ak+1)^2)*t1(j+1)));
                Call = blsprice(S1(i+1),E,rr,T1(j+1),sigma);
                S(l,7) = Call;
                l=l+1;
        end;
end;
for l = 1:k
        S(l,8) = S(l,7) - S(l,6);
end;

csvwrite('call4.dat',S);

echo off
```

C Appendix: Matlab code European put solution

```
echo on
format short

% steps of x and t, m and n respectively
m=133;
n=200;

% maximum values of indices considering +1
M=m+1;
N=n;

% Upper and lower boundaries of x and t
% respectively and range of x and t respectively
bx1=log(12.5/10)-(72*0.0225);
bx2=log(12.5/10)+(62*0.0225);
bt1=0;
bt2=0.045;
rx=bx2-bx1;
rt=bt2-bt1;

% steps of x and t and values of x and t for each step
ix=rx/M;
it=rt/N;
x1=bx1:ix:bx2;
t1=bt1:it:bt2;

%Black Scholes Vars
rr=0.04;
E=10;
sigma=0.3;
ak=rr/((sigma^2)/2);
S1 = E*exp(x1);
T1 = (t1/((sigma^2)/2));

% heat equation constants
w=1;
y=(it*w^2)/(ix^2);
z=y/2;

% dimensions for matrix
k=m*n;

% gcd and lcm
g1 = gcd(m,n);
l1 = lcm(m,n);
```

```
% index beginning
j=0;
i=1;
l=1;
f=0;

% make matrix and vector for the equations
Q = zeros(k,1);
A = sparse(k,k);

% populate matrix
while l <= k
        if mod(l,l1) == 1 & l>l1
                j = 0 + f;
        elseif mod(l,l1) ~= 1 & j == N
                j = 0;
        end;
        if l <= k & i == M
                i = 1;
        end;
        J = [j+1,i,(1+y);j,i,(y-1);j+1,i+1,-z;j+1,i-1,-z;...
        j,i+1,-z;j,i-1,-z];
        for p = 1:6
                if J(p,1) == 0
                        Q(l,1) = Q(l,1) + max((exp(0.5*...
                        (ak-1)*x1(J(p,2)+1))-exp(0.5*...
                        (ak+1)*x1(J(p,2)+1))),0)*(-1*J(p,3));
                elseif J(p,1) ~= 0 & J(p,2) ~= 0 & J(p,2) ~= M
                        A(l,(J(p,1)-1)*m + J(p,2)) = J(p,3);
                elseif J(p,1) ~= 0 & J(p,2) == 0
                        Q(l,1) = Q(l,1) + (exp(-ak*t1(J(p,1)+1))...
                        *(exp((0.5*(ak-1)*x1(J(p,2)+1))...
                        +(0.25*((ak+1)^2)*t1(J(p,1)+1)))))...
                        *(-1*J(p,3));
                elseif J(p,1) ~= 0 & J(p,2) == M
                        Q(l,1) = Q(l,1) + 0;
                end;
        end;
        j=j+1;
        i=i+1;
        if gcd(m,n) > 1 & mod(l,l1) == 0
                f=f+1;
        end;
        l=l+1;
end;

% solve and display results vector
S = zeros(k,9);
S(:,5) = (A\Q);
l=1;
```

```
i=int32(1);
j=int32(1);
for j = 1:n
        for i = 1:m;
                S(l,1) = j;
                S(l,2) = i;
                S(l,3) = T1(j+1);
                S(l,4) = S1(i+1);
                S(l,6) = S(l,5)*E*exp((-0.5*(ak-1)*x1(i+1))...
                +(-0.25*((ak+1)^2)*t1(j+1)));
                [Call,Put] = blsprice(S1(i+1),E,rr,T1(j+1),sigma);
                S(l,7) = Put;
                l=l+1;
        end;
end;
for l = 1:k
        S(l,8) = S(l,7) - S(l,6);
        if S(l,7)>(10^(-3))
                S(l,9) = S(l,8)/S(l,7);
        end;
end;

csvwrite('put.dat',S);

echo off
```

References

[1] Implementing finite difference solvers for the black-scholes pde, 2009. URL `http://faculty.baruch.cuny.edu/lwu/890/SpruillOnFiniteDifferenceMethod.doc`.

[2] F. Black and M. Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81:637–654, 1973.

[3] M. P. Coleman. *An Introduction To Partial Differential Equations with Matlab.* Chapman & Hall/CRC, 2005.

[4] M. Cooney. Report on the accuracy and efficiency of the fitted methods for solving the black-scholes equation for european and american options, November 2000. URL `http://www.datasimfinancial.com/UserFiles/articles/BlackScholesReport2.ps`.

[5] C. F. Gerald and P. O. Wheatley. *Applied Numerical Analysis.* Addison Wesley, 3 edition, 1984.

[6] J. C. Hull. *Options, Futures, and other Derivatives.* Pearson Prentice Hall, 7 edition, 2009.

[7] P. Wilmott, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives.* Cambridge University Press, 2008.