MATH3423

RESEARCH PROJECT

KAREEM POWELL

620098669

April 23, 2019

# Option Valuation using Finite Difference Methods

# Acknowledgment

I would like to express my gratitude to my course supervisor, Dr. Nordia Thomas, whose support was of tremendous help over the course of this project. I would also like to thank Dr. Thomas for her comments as they greatly improved this manuscript.

I would also like to thank my peers for the insight and feedback that was provided throughout the scope of this research project.

## Abstract

The finite difference method is a mathematical construct that can be used to solve partial differential equations. In this study, we used the finite difference method to solve the Black-Scholes-Merton partial differential equation to calculate options prices. Three methods were used: the Implicit Method, the Explicit Method, and the Crank-Nicolson Method. Using some code and the help of MATLAB I was able to calculate for each of the three methods listed above the values of both call and put options using the Black-Scholes-Merton partial differential equation. Furthermore, the Binomial Cox-Ross-Rubinstein Model was introduced briefly to conduct a comparative study using this model and the finite difference methods. An analysis was carried out to ascertain which of the above methods would agree with the Black-Scholes value of an option. It was found that only the Explicit Method, the Implicit Method, and the Binomial CRR Model produced similar values. A second analysis was done to compare which of the models would converge to the Black-Scholes value of an option given that the number of time intervals L and the intervals of the stock prices were increased. It was found that the Crank-Nicolson method converged faster than the Binomial Model. Hence, we conclude that the finite difference model is more appropriate than the Binomial CRR model.

Table of Contents

# INTRODUCTION

An option is a financial instrument that gives its holder the right but not the obligation to either buy or sell a predetermined asset at a given time in the future. The individual who issues the option is referred to as the writer of the option. There are two types of options, puts and calls. A put option gives the holder of the option the right but not the obligation to sell a predetermined asset at a given point in time. On the other hand, a call option gives the holder the right but not the obligation to purchase a predetermined asset at a given point in time. Options are normally held till their expiration or maturity date. The price at which the asset will be sold or bought at the maturity date is called the strike price. There are two other types of options, European or American options. These types of options defer not regarding their payoff but their expiration date. European options can be exercised at any time up to the expiration date. However, American options can only be exercised at the expiration date.

Options can be used both for speculating and hedging. Speculating is when a person takes a position in an asset to make a return based on its up or downward movement. For example, if an investor believes that the shares of a company will rise, he or she can purchase a call option; therefore, making a profit if the share price increases. If the share price does not increase, then the investor would decide against exercising the option. Hedging, on the other hand, allows individuals to offset the risk of their investment through the purchase of options. In this case, options allow individuals to be able to hedge the risk of that given portfolio or investment. The person who is usually on the other side of a hedge is the speculator. For example, security holders can hedge their risk if they believe that the price of their securities will decrease in the future. This can be done by purchasing a put option.

Options are traded on various exchanges. Such exchanges include the Chicago Board Options Exchange (CBOE), NASDAQ OMX, NYSE Euronext, the International Securities Exchange, and the Boston Options Exchange. Options are also traded between financial institutions and these exchanges are referred to as over-the-counter (OTC deals). Currently, options are traded on more than 2,500 different stocks. One option contract can give its holder the right to buy or sell 100 shares. To buy and sell options, a value must be assigned to them. The assigning of these values is referred to as option valuation and is done with the use of various models. This research contains an overview of finite difference methods and their applications in option pricing. We will be looking at the application of this method along with the application of the Binomial Method. Both models apply mathematical principles and formulas to calculate the prices of both call and put options.

## 1. <u>The Finite Difference Method</u>

The finite difference method of option pricing involves the use of mathematical applications to approximate an option price's differential equations using sets of discrete-time difference equations. The three finite difference methods that this study will cover include the explicit method, the implicit method, and the Crank-Nicolson Method.

Each finite difference method has four steps.

1. The first step involves converting the partial differential equation into a discrete-time differential equation.
2. Specify a grid of potential current and future prices of the asset that is being evaluated.
3. Calculate the payoff of the option given the prices within a certain range or boundary of the grid.
4. Iteratively calculate the payoff of all other options including the point of the underlying price (base year) and the current price. The procedure to iteratively calculate the payoffs is different for each finite difference method. That is for the Explicit, Implicit, or Crank Difference Methods the valuation of the asset will utilize a different method.

**Partial Differential Equations**

To introduce the topic of finite difference methods, we shall first refer to partial differential equations. A partial differential equation can be of the form:

$$a(t, x,y)U_t + b(t,x,y)U_x + (t,x,y)U_{yy}=f(t,x,y) \qquad (1.0)$$

Where

1. t,x, and y are the independent variables
2. a,b,c, and f are known functions of the independent variables

3. U is the dependent variable and is an unknown function of the independent variables.

4. $U_t$, $U_x$ and $U_{yy}$ are partial derivatives where $U_t=(\partial U/\partial t)$, $U_x=(\partial U/\partial x)$ and $U_{yy}=(\partial^2 U\backslash\partial y^2)$

Solving a PDE involves finding the unknown function U; which is the exact solution that satisfies both the initial and boundary conditions. Some Partial Differential Equations do not have exact solutions so in these cases, an approximate solution to the exact solution U must be found. These approximations are usually found numerically using computer software. These approximations are then used as replacements for the partial derivatives within the PDE. From this newly defined equation, an approximate solution can then be found.

The finite difference method works by replacing the region over which the independent variables of the partial differential equations stretch, with a mesh of points that are approximated using Taylor's theorem.

Given the equation (1.1) from above,

$$U(x_0+h)=U(x_0)+hU_x(x_0)+h^2U_{xx}(x_0)/2!+....+h^{n-1}U_{n-1}(x_0)/(n-1)!+O(h^n), \quad (1.1b)$$

where,

1. $U_x=dU/dx$, $U_{xx}=d^2U/dx^2$,....., $U_{n-1}=d^{n-1}U/dx^{n-1}$

2. $U_x(x_0)$ is the derivative of U concerning x evaluated at $x=x_0$.

3. And $O(h^n)$ is an unknown error term.

Our first step will be to reinterpret the equation given in (1.1b). To achieve this, we will truncate the right side of the equation which will yield the error term $O(h^n)$.

## Simple Finite Difference Methods

Truncating after the first derivative we have:

$U(x_0+h)=U(x_0) + hU_x(x_0) + O(h^2)$

(1.2)

Rearranging this equation, we have:

$U_x(x_0)=\{U(x_0+h)-U(x_0)\}/h + O(h^2)/h$

$=> \quad U_x(x_0)=\{U(x_0+h)-U(x_0)\}/h + O(h)$ (1.3)

Removing the O(h) term we have

$U(x_0)=\{U(x_0+h)-U(x_0)\}/h$

which is the first-order approximation to the partial differential equation in (1.1b). The approximation is called a first-order approximation because we start at point $x_0$ and move one step in the direction of $x_0+h$. The step size is known as h, where h>0.

**Constructing a Finite Difference Toolkit**

Suppose U is a function of two variables t and x; i.e., U(t,x), where if t is constant U now becomes a function of one variable x. Furthermore, suppose we replace the step-variable h with Δx.

Therefore, from equation (1.1b) we have

$U(t,x+\Delta x)=U(t,x_0)+\Delta xU_x(t,x_0)+(\Delta x^2/2!)U_{xx}(t,x_0)+(\Delta x^3/3!)U_{xxx}(t,x_0)+..+\{\Delta x^{(n-1)}/(n-1)\}U_{n-1}(t,x_0)+O(\Delta x^n)$

(1.4)

Removing up to $O(\Delta x^2)$ gives,

$U(t, x_0+\Delta x)= U(t,x_0) + \Delta xU_x(t,x_0) + O(\Delta x^2)$

(1.5)

$=> U_x(t,x_0)=\{U(t,x_0+\Delta x)-U(t,x_0)\}/\Delta x -O(\Delta x^2)/\Delta x$

$=>U_x(t,x_0)=\{U(t,x_0+\Delta x)-U(t,x_0)\}/\Delta x - O(\Delta x)$ (1.6)

Equation (1.6) holds for any values of the $(t,x_0)$. Also, for numerical schemes for PDEs, we are restricted to a grid of discrete x values, $x_1$, $x_2$, .., $x_N$, and discrete t levels $0=t_0,t_1,t_2$... Furthermore, suppose that we use constant grid spacing such that $x_i=x_{i-1}+\Delta x$. Analyzing equation (1.6) on a grid point $(t_n, x_i)$ we have

$U_x(t_n,x_i)=\{U(t_n,x_i+\Delta x)-U(t_n,x_i)\}/\Delta x - O(\Delta x)$ (1.7)

Using common subscript notation, we have:

$U^n_i=U(t_n,x_i)$ (1.8)

So, equation (1.7) becomes

$U_x(t_n,x_i)\approx(U^n_{i+1}-U^n_i)/\Delta x$ (1.9)

Which is a first-order forward difference approximation to $U_x(t^n, x_i)$.

Furthermore, using equation (1.6) we derive another Finite Difference Approximation.

Replacing $\Delta x$ with $-\Delta x$ we have

$U(t, x_0-\Delta x)= U(t,x_0) - \Delta x U_x(t,x_0) + O(\Delta x^2)$

(1.10)

$=>U_x(t,x_0)=\{U(t,x_0)-U(t,x_0-\Delta x)\}/\Delta x -O(\Delta x^2)/\Delta x$

$=>U_x(t,x_0)=\{U(t,x_0)-U(t,x_0+\Delta x)\}/\Delta x - O(\Delta x)$

Therefore,

Using the previous notation of $U^n_i\approx U(t_n, x_i)$ and evaluating at points $(t_n, x_i)$ we have,

$U_x(t_n,x_i)\sim(U^n_i - U^n_{i-1})/\Delta x$

Which is the first-order backward approximation to $U_x(t_n, x_i)$.

By increasing the order of the Taylor approximation we can find a more accurate approximation of the partial differential equation. To find a second-order central difference approximation for $U_x(t_n, x_i)$ we truncate equation (1.4) to $O(\Delta x^3)$, subtract this new expression from equation (1.4) and then evaluate this equation at the points $(t^n, x_i)$.

Starting from the initial equation (1.4) we have:

$U(t,x+\Delta x)=U(t,x_0)+\Delta x U_x(t,x_0)+(\Delta x/2!)U_{xx}(t,x_0)+(\Delta x/3!)U_{xxx}(t,x_0)...+\{\Delta x^{n-1}/(n-1)\}U_{n-1}(t,x_0)+O(\Delta x^n)$

(1.11)

Removing up to $O(\Delta x^3)$ gives:

$U(t, x_0+\Delta x)= U(t,x_0) + \Delta x U_x(t,x_0) +(\Delta x/2!)U_{xx}(t,x_0) + O(\Delta x^3)$ (1.12)

Subtracting equation (1.12) from equation (1.4), evaluating at the points $(t_n, x_i)$ and proceeding with the appropriate substitutions we get,

$U_x(t_n,x_i)\approx(U^n_{i+1} - U^n_{i-1})/2\Delta x$

(1.13)

which is the second-order central difference finite difference approximation to the equation $U_x(t_n, x_i)$.

In addition to the second-order central difference FD approximation, there is also the second-order symmetric difference FD approximation to $U_x(t_n, x_i)$.

Truncating equation (1.4) to $O(\Delta x^4)$ and doing the necessary steps we get,

$U_{xx}(t_n,x_i)\approx(U^n_{i+1} - 2U^n_i+ U^n_{i-1})/(\Delta x^2)$.

(1.14)

The above results can be used to form a finite difference toolkit with the partial derivatives concerning x.

Similarly, a finite difference toolkit can be formed in terms of t. That is, for a forward difference FD approximation we have $U_t(t_n, x_i) \approx (U^{n+1}_i - U^n_i)/\Delta t$. (1.15)

Secondly, for a first-order backward difference FD approximation, for an equation, we have

$U_x(t_n, x_i) \approx (U^n_i - U^{n-1}_{i-1})/\Delta x$.

(1.16)

For a second-order central difference FD approximation for an equation concerning t, we have

$U_t(t_n,x_i) \approx (U^{n+1}_i - U^{n-1}_i)/2\Delta t$ . (1.17)

And lastly, for a second-order symmetric FD approximation for an equation concerning t, we have

$U_{tt}(t_n,x_i) \approx (U^{n+1}_i - 2U^n_i + U^{n-1}_i)/\Delta t^{2.}$

(1.18)

## 2. The Finite Difference Method and its Applications to Options

The finite difference method comprises three methods; the implicit method, the explicit method, and the Crank-Nicolson method. Each of these finite difference methods is comprised of four steps:

1. Discretizing the differential equation
2. Specifying a grid of current potential and future prices for the underlying asset. This grid ranges from the minimum price of the underlying to its maximum price from today to the maturity date of the option.
3. Calculating the option payoff at certain points on the boundaries of the grid.
4. Iteratively determining the price of the option at the various underlying prices.

The steps that have been described above are common to each of the three methods of option pricing. The only difference lies in the iterative valuation of the option prices within the points of the defined grid.

One partial differential equation that describes the price of options about an underlying asset is the Black-Scholes-Merton partial differential equation. This equation combines the value of an asset S with an option with price f(s,t).

The equation is denoted as follows:

$$\frac{\partial F}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 \mathrm{S}^2 \frac{\partial^2 \mathrm{f}}{\partial S^2} = rf$$

(2.0)

One method of solving this equation is to use one of the three finite difference methods mentioned above. This can be done by substituting the partial derivatives with the appropriate difference equations. The difference equation that is used differs based on the finite difference method that is being employed. That is, the implicit, explicit, and Crank-Nicolson methods all use different combinations of difference approximations to find the solution to the above PDE. The finite difference method is used whenever the exact solution to the partial differential equation cannot be found.

Using the approximate solutions to equation $U(t_n, x_i)$ from the previous section; however, shifting from a two-variable function to a one-variable function and representing U in terms of a different function *f* we have:

The first-order forward difference FD approximation:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \tag{2.1}$$

The first-order backward difference FD approximation:

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \tag{2.2}$$

The central difference FD approximation:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \tag{2.3}$$

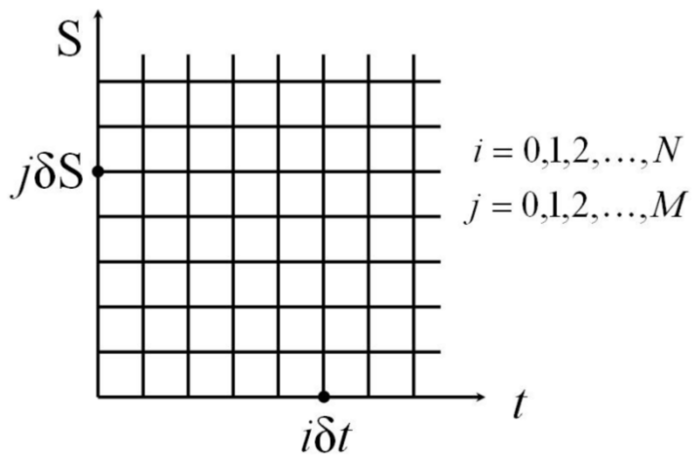
The second-order approximation:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$$

(2.4)

Our next step in discretizing the partial differential equation in (2.0) would be to replace the partial derivatives with our approximations.

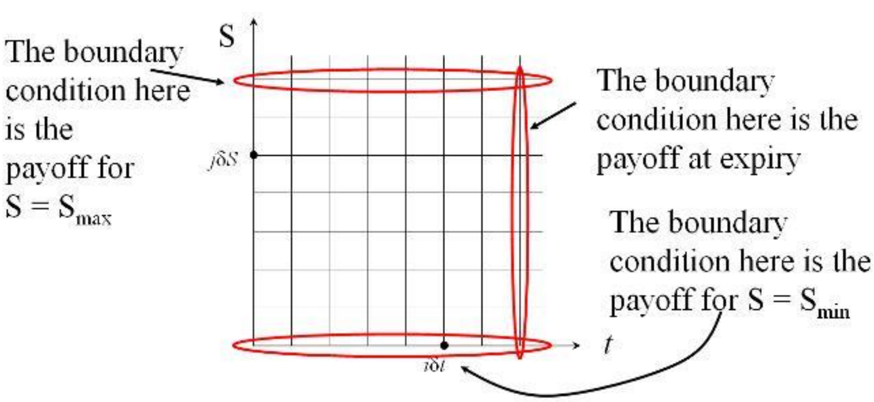
The next step in valuing options using the finite difference method is to specify a grid of the potential future values of the given underlying asset prices. This is done by plotting a graph of the underlying asset price against very small movements in time. Consider the grid as follows:




As is shown in the diagram, the time to expiration is divided into N equal intervals, where the expiration date is t. Furthermore, the initial asset value to the maximum asset value is divided into M equal intervals. Therefore, we are left with a grid made up of N+1 levels of time and M+1 prices.

The third step as discussed above is to determine the value of the option at the boundaries of the grid.

Consider the diagram below,



To determine the value of the option at the boundaries of the grid, we consider when the underlying asset S has price $S_{max}$ when the underlying asset S has price $S_{min}$ and lastly the value of the underlying asset at the expiry date.

## 3. The Implicit Finite Difference Method

For the Implicit Difference Method, we shall discretize the Black-Scholes-Merton partial differential equation

$$\frac{\partial F}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

(3.0)

using the approximations given above. Please note that the Black-Scholes-Merton Equation is a formula linking the value of an option to the value of an underlying asset. We shall discretize the Black-Scholes-Merton Equation is discretized using the following formulas:

The first-order forward difference approximation for $F_{t:}$

$$\frac{\partial f}{\partial t} = \frac{f_{i+1,j} - f_{i,j}}{\delta t}$$

(2.4)

The first-order central difference FD approximation for $F_{s:}$

$$\frac{\partial f}{\partial S} = \frac{f_{i,j+1} - f_{i,j-1}}{2\delta S}$$

(2.5)

And the standard approximation for $F_{ss:}$

$$\frac{\partial^2 f}{\partial S^2} = \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{(\delta S)^2}$$

(2.6)

Please note that I denote steps in time, while j denotes steps in the price of the underlying asset.

Therefore, the Black-Scholes-Merton formula using the above approximations is:

$$\frac{f_{i+1,j} - f_{i,j}}{\delta t} + rj\delta S\,\frac{f_{i,j+1} - f_{i,j-1}}{2\delta S} + \frac{1}{2}\sigma^2 (j\delta S)^2\,\frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{(\delta S)^2} = rf_{ij}$$

(2.7)

Which eventually reduces to,

$a_j{*}f_{i,j-1} + b_j{*}f_{i,j} + c_j{*}f_{i,j+1} = f_{i+1,j}$

(2.8)

Where,

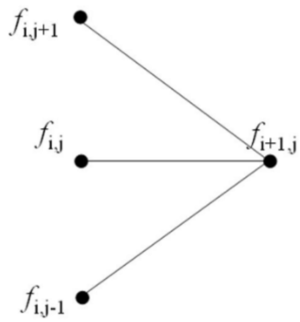
$a_j{*}=½\ \delta t(rj-\sigma^2 j^2)$

$b_j{*}=1+\delta t(\sigma^2 j^2+r)$

$c_j{*}=½\ \delta(-rj-\sigma^2 j^2)$

From the above equations, it is then known that the value of three different options at time t is dependent upon the value of the underlying asset at time t+1.

See the following diagram:




This diagram illustrates what was discussed above, where the option prices of the underlying asset price are dependent on the price of the option at time i+1. The unknown values of the option are being considered at prices j-1,j, and j+1 and are dependent on or can be found using the option price at time i+1. Therefore, using the value of the option at expiry we can then find the price of an option at Δt before. Consequently, we can find the price of the option at 2Δt before expiry using the price of the option at Δt. Working backward, we can eventually find the price of the option today.

The above equations can be represented in matrix form. That is,

BFi=Fi+1+Ki where i is from N-1 to 0 (2.9)

Where

$$F_i = \begin{bmatrix} f_{i,1} \\ f_{i,2} \\ \vdots \\ \vdots \\ f_{i,M-1} \end{bmatrix}$$

$$K_i = \begin{bmatrix} -a_1 f_{i,0} \\ 0 \\ \vdots \\ 0 \\ -c_{M-1} f_{i,M} \end{bmatrix}$$

And

$$B = \begin{bmatrix} b_1^* & c_1^* & 0 & \cdots & 0 & 0 \\ a_2^* & b_2^* & c_2^* & \cdots & 0 & 0 \\ & a_3^* & b_3^* & \cdots & 0 & 0 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & \cdots & a_{M-1}^* & b_{M-1}^* \end{bmatrix}$$

We shall use the matrix form of the equation to solve for the option prices.

**Assumptions for the Implicit Finite Differencing Method**

To properly calculate options prices, it is important to ensure that the matrix equation in (2.9) is stable. Stability is ensured when the infinity norm of the matrix B^-1 is less than or equal to 1. This will enable the successive values of equation Fi to converge to a number less than infinity.

See the following code for an example of how to price options using the Implicit Finite Difference Method:

**Example 3.1**

Consider pricing a European Call and Put option with the following parameters, X = \$60, $S_0$ = \$50, r = 5%, σ = 0.2, and T = 1.

The Black-Scholes price for the Call option is $1.624, and the Put option is $8.697

See the following MATLAB implementation for the Implicit FDM method:

```
>> cPrice = finDiffImplicit(60,50,0.05,0.2,0:1:100,0:0.01:1,'CALL')

cPrice=1.5826

>> pPrice = finDiffImplicit(60,50,0.05,0.25,0:1:100,0:0.01:1,'PUT')

pPrice=8.698
```

**4. The Explicit Method**

For the explicit method, we shall use the Black-Scholes-Merton partial differential equation,

$$\frac{\partial F}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf \quad (4.0)$$

to derive a way to price an option. This will be done by approximating the values of the partial derivatives using the FD approximations. Using the finite difference approximations derived above we have:

The first-order backward FD approximation for $F_t$:

$$\frac{\partial f}{\partial t} = \frac{f_{i,j} - f_{i-1,j}}{\delta t} \quad (4.1)$$

The first-order central difference FD approximation for $F_s$.

$$\frac{\partial f}{\partial S} = \frac{f_{i,j+1} - f_{i,j-1}}{2\delta S}$$

(4.2)

And the standard approximation for $F_{ss}$.

$$\frac{\partial^2 f}{\partial S^2} = \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{(\delta S)^2}$$

(4.3)

Where the indices i represent points in time while the indices j represent points in the stock prices.

Replacing the partial differential equations with the appropriate approximations we have

$$\frac{f_{i,j} - f_{i-1,j}}{\delta t} + rj\delta S \frac{f_{i,j+1} - f_{i,j-1}}{2\delta S} + \frac{1}{2}\sigma^2 (j\delta S)^2 \frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{(\delta S)^2} = rf_{ij}$$

(4.4)

Which eventually reduces to:

$f_{i-1,j}=a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1}$

(4.5)

Where,

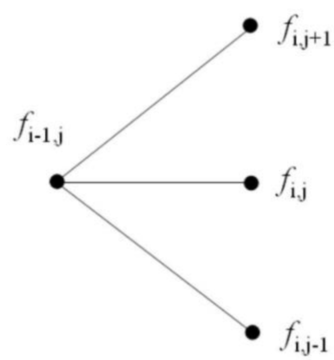
$a_j=½\delta t(\sigma^2 j^2-rj)$

$b_j=1+\delta t(\sigma^2 j^2+r)$

$c_j=½\ \delta(\sigma^2 j^2+rj)$

From the above equations, given the option price at time t, we can then calculate the value of the option price at time t-1. Similarly to the implicit method, we can then find the value of the option given the price/prices at a given point in the future. The difference between the implicit and explicit finite difference methods is that for the explicit method, we use values of the option in the future to calculate the prices of options for different underlying asset prices. On the other hand, for the implicit method, we used the value of the option at the expiry date to find option prices for their respective underlying asset prices.

What was described above can be seen in the following diagram:




Where the values of the option at time i-1 are ascertained from the values of the options at their respective asset prices. Furthermore, this diagram implies that given option payoffs at the expiry date we can then find the value of the option Δt before expiry. Furthermore, by working backwards we can then find the value of the option at time t=0; which is today's price.

A matrix formulation for the above equation (3.4) can be expressed as follows:

$F_{i-1}=AF_i+K_i$ where i is from N to 1.

(4.6)

Where equation (3.5) can be expressed as

$$F_i = \begin{bmatrix} f_{i,1} \\ f_{i,2} \\ \vdots \\ \vdots \\ f_{i,M-1} \end{bmatrix}$$

$$K_i = \begin{bmatrix} a_1 f_{i,0} \\ 0 \\ \vdots \\ 0 \\ c_{M-1} f_{i,M} \end{bmatrix}$$

$$A = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & \cdots & 0 & 0 \\ & a_3 & b_3 & \cdots & 0 & 0 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & \cdots & a_{M-1} & b_{M-1} \end{bmatrix}$$

Equation (4.6) will be used to solve for option prices.

**Assumptions for the Explicit Finite Difference Method**

To solve the matrix equation in (4.6), we need to verify that the equation is stable. Stability can be verified by finding the infinite norm of matrix A. If the infinite norm of the matrix A is less than or equal to 1 then the equation is stable. That is, the LHS of the equation converges to a solution because the successive values of $F_i$ will get smaller and smaller, therefore converging.

See the code below for an illustration of how to solve the above matrix equation using MATLAB.

**Example 4.1**

Consider pricing a European Call and Put option with the following parameters, X = \$60, $S_0$ = \$50, r = 5%, σ = 0.2, and T = 1.

The Black-Scholes price for the Call option is \$1.624, and the Put option is \$8.697

The implementation of the Explicit FDM using MATLAB is as follows:

```
>> cPrice = finDiffExplicit(60,50,0.05,0.2,0:1:100,0:0.001:1,'CALL')

cPrice=1.621

>> pPrice = finDiffExplicit(60,50,0.05,0.2,0:1:100,0:0.001:1,'PUT')

pPrice=8.695
```

## 5. The Crank-Nicolson Method

The Crank-Nicolson Method is based on both the implicit method and the explicit method. It represents the average between the two methods.

Consider the grid below,

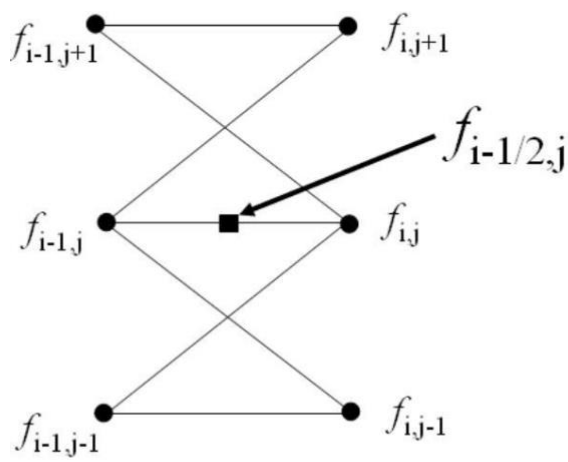



Figure 5.1: Grid of the Price Points for the Crank-Nicolson Method

In the explicit method the price of the option at $f_{i-1,j}$ is found using the values of the option at nodes $f_{i,j+1}$, $f_{i,j}$ and $f_{i,j-1}$. Whilst in the implicit method the values of the option at nodes $f_{i-1,j+1}$, $f_{i-1,j}$ and $f_{i-1,j-1}$ are found using the option value at node $f_{i,j}$. However, in the Crank-Nicolson method, the option values of the nodes on the left side are based on each node on the right side. That is the prices of the option at nodes $f_{i-1,j+1}$, $f_{i-1,j}$ and $f_{i-1,j-1}$ are based on the option values at the nodes $f_{i,j+1}$, $f_{i,j}$ and $f_{i,j-1}$. To derive the Crank-Nicolson equations we consider the price of the option at the node $f_{i-1/2,j}$. (Please note that a price will not be computed for this node. On the contrary, this point will be used for notational purposes and will not appear in the final equation.)

Consider the Black-Scholes-Merton partial differential equation. We aim to discretize this equation:

$$\frac{\partial F}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

(5.1)

Using the following approximations:

1. The Central Approximation for $F_t$.

$$\frac{\partial f_{i-1/2,j}}{\partial t} = \frac{f_{i,j} - f_{i-1,j}}{\delta t} + o(\delta t^2)$$

(5.2)

2. A Central Approximation for $F_s$.

$$\frac{\partial^2 f_{i-1/2,j}}{\partial S^2} = \frac{1}{2}\left[\frac{\partial^2 f_{i-1,j}}{\partial S^2} + \frac{\partial^2 f_{i,j}}{\partial S^2}\right]$$
$$= \frac{1}{2}\left[\frac{f_{i-1,j+1} - 2f_{i-1,j} + f_{i-1,j-1}}{\partial S^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\partial S^2}\right] + O(\delta S^2)$$

(5.3)

3. And using a standard approximation for Fss.

$$\frac{\partial^2 f_{i-1/2,j}}{\partial S^2} = \frac{1}{2}\left[\frac{\partial^2 f_{i-1,j}}{\partial S^2} + \frac{\partial^2 f_{i,j}}{\partial S^2}\right]$$
$$= \frac{1}{2}\left[\frac{f_{i-1,j+1} - 2f_{i-1,j} + f_{i-1,j-1}}{\partial S^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\partial S^2}\right] + O(\delta S^2)$$

(5.4)

Substituting these equations into the Black-Scholes-Merton equation and reducing it we have:

$$-\bar{a}_j f_{i-1,j-1} + (1-\bar{b}_j) f_{i-1,j} - \bar{c}_j f_{i-1,j+1} = \bar{a}_j f_{i,j-1} + (1+\bar{b}_j) f_{i,j} + \bar{c}_j f_{i,j+1} \quad (5.5)$$

where:

$$\bar{a}_j = \frac{\delta t}{4}\left(\sigma^2 j^2 - rj\right)$$
$$\bar{b}_j = -\frac{\delta t}{2}\left(\sigma^2 j^2 + r\right)$$
$$\bar{c}_j = \frac{\delta t}{4}\left(\sigma^2 j^2 + rj\right) \quad (5.6)$$

The above equation (5.2), when expanded for all values of i and j, leads to M-1 equations. Therefore we can then solve these simultaneous equations to calculate the value for f at each node. To solve the above we shall use the matrix representation as found below:

Consider the matrix equation below:

$$CF_{i-1}=DF_i+K_{i-1+}K_i$$

Where i is from N to1.

Where

$$F_i = \begin{bmatrix} f_{i,1} \\ f_{i,2} \\ \vdots \\ \vdots \\ f_{i,M-1} \end{bmatrix}$$

$$K_i = \begin{bmatrix} \bar{a}_1 f_{i,0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \bar{c}_{M-1} f_{i,M} \end{bmatrix}$$

$$C = \begin{bmatrix} 1-\bar{b}_1 & -\bar{c}_1 & 0 & \cdots & 0 & 0 \\ -\bar{a}_2 & 1-\bar{b}_2 & -\bar{c}_2 & \cdots & 0 & 0 \\ & -\bar{a}_3 & 1-\bar{b}_3 & \cdots & 0 & 0 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & \cdots & -\bar{a}_{M-1} & 1-\bar{b}_{M-1} \end{bmatrix}$$

And

$$D = \begin{bmatrix} 1+\bar{b}_1 & \bar{c}_1 & 0 & \cdots & 0 & 0 \\ \bar{a}_2 & 1+\bar{b}_2 & \bar{c}_2 & \cdots & 0 & 0 \\ & \bar{a}_3 & 1+\bar{b}_3 & \cdots & 0 & 0 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & \cdots & \bar{a}_{M-1} & 1+\bar{b}_{M-1} \end{bmatrix}$$

**Assumptions for the Crank-Nicolson Method**

A necessary assumption that is required to solve equation (5.1) is for both stability and convergence. These properties are necessary to ensure that numerical solutions to these equations can be found. That is, they converge on a solution. See the following equation:

$$\left\| C^{-1} D \right\|_\infty \leq 1$$

That is the infinite norm of the inverse of C times the matrix D is less than equal to 1. When this equation holds the successive values of $F_i$ get smaller and smaller and therefore the algorithm converges

**See the implementation of the following code for an example of the Crank-Nicolson method.**

**Example 5.1**

Consider pricing a European Call and Put option with the following parameters, X = \$60, $S_0$ = \$50, r = 5%, σ = 0.2, and T = 1.

Implementing the formula for the Crank-Nicolson method in MATLAB we have:

```
>> cPrice=finDiffCN(60,50,0.5.0:1:100,0:0.1:1,'Call')

cPrice=1.6216

>> pPrice=finDiffCN(60,50,0.5,0.2,0:1:100,0:0.01:1,'PUT')

pPrice=8.6952
```

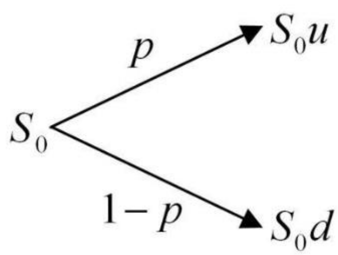
## 5. The Binomial Method

Binomial Models are by far the simplest methods of option pricing. Within this method there are three main steps involved in pricing options:

1. Calculate the potential future price of the underlying asset at expiry.
2. Calculate the option payoff at expiry for each of the underlying asset prices.
3. Discount the payoffs back to today to determine the value of the option today.

The first step in creating the binomial model is to create a true containing the potential future prices of the underlying asset. This can be achieved by using risk-neutral pricing, constructing a one-step binomial model, and using a multi-period binomial tree.

Consider the One-Step Binomial Model below:




From the above diagram, we have the following:

- $S_0$: The Price of the Stock today
- p: The probability of a rise in the stock price.
- u: The factor by which the stock price rises
- d: The factor by which the stock price falls.

Assuming that the price of an asset today is $S_0$ given a step of Δt in the future we have that the price of the stock can either be $S_u$ ($S_0u$) or $S_d$ ($S_0d$) where the $S_u$ is the price of the stock given that it rises over the interval Δt and $S_d$ is the price of the stock given that it decreases over the time interval Δt. The stock price $S_0$ follows that of a random walk and the probability that this price will rise over the given interval is p. Similarly, the probability that the stock price will fall is 1-p.

**One-step Binomial Model**

For the one-step binomial model, three main equations are used to solve for the parameters p, u, and d where:

$$pu + (1-p)d = e^{r\Delta t} \quad (6.1)$$

Equation (6.1) arises out of the requirement that the expected return of the binomial model is equal to the expected return in a risk-neutral world.

The second equation,

$pu^2 + (1-p)d^2 - e^{r\Delta t}$=sigma squared$\Delta t$

(6.2)

ensures that the variance in a risk-free world is equal to the variance of the binomial model.

The third equation was devised by Cox, Ross, and Rubinstein. That is,

u=1/d

(6.3)

Rearranging the parameters above we get:

$$p = \frac{e^{r\Delta t} - d}{u - d}$$

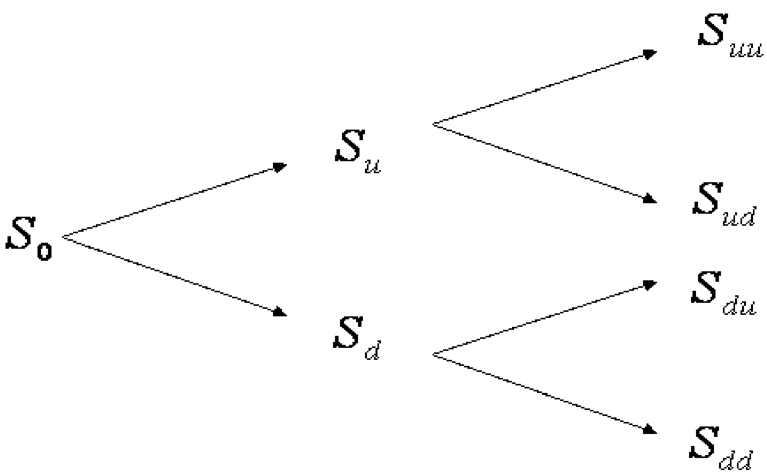
$$u = e^{\sigma\sqrt{\Delta t}}$$

$$d = e^{-\sigma\sqrt{\Delta t}}$$

These formulas are the equations for the Cox-Ross Rubinstein model.

**The Multi-Step Binomial Model**

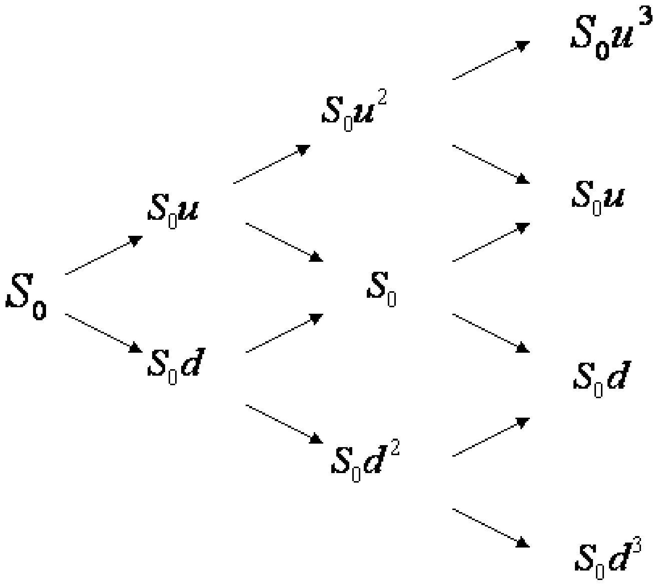
Consider the two-step binomial model below:




In the above model, the initial price of the stock is $S_0$. The stock price can either move up to $S_u$ or down to $S_d$ in the first period. In the second period, the stock price can increase from $S_u$ to $S_{uu}$ or decrease from $S_u$ to $S_{ud}$. Additionally, the price can rise from $S_d$ to $S_{du}$ or fall from $S_d$ to $S_{dd}$. If $S_{ud}=S_{du}$ then the tree is said to be recombining. If not then the tree is non-recombining. Because options prices are normally calculated over hundreds of thousands of periods, non-recombining trees are not usually calculated in practice due to financial constraints or a lack of resources.

Equation (6.3) ensures that the CRR binomial model generates a recombining tree.

A multi-step binomial model can be represented by the below:




The period from $S_0$ to node $S_0u^3$ is the time from today to the expiry of the option. This period is split between the nodes to allow the calculation of the potential future stock prices at the various nodes. That is the tree contains all the potential future values of the stock from today to its maturity date.

**Calculating the Option Payoffs at Maturity**

The next step in the binomial pricing model is to calculate the payoffs of the options. The payoffs for a simple put and call are:

Put: $V_N=\max(X-S_N,0)$

Call: $V_N=\max(S_N-X,0)$

where,

- N designates a node at expiry
- $V_N$ is the option value
- X is the strike price
- $S_N$ is the value of the underlying asset

**Discounting the Option Payoffs**

The last step in the Binomial Pricing model is to discount the option values back to today. This can be done using a method called backwards induction which involves starting from the expiry date and working sequentially back in time to calculate the option prices at each of the nodes. Using the appropriate formula the result follows:

European Put or Call: $Vn= e^{-r\Delta t}(pV_u+(1-p)V_d)$

American Put: $V_n = \max(X-S_n,e^{-r\Delta t}(pV_u+(1-p)V_d))$

American Call: $V_n = \max(S_n-X,e^{-r\Delta t}(pV_u+(1-p)V_d))$

where

- n designates a node before expiry.
- $V_n$ is the option value.
- X is the strike.
- $S_n$ is the price of the underlying asset.
- p is the probability of an upward price movement.
- $V_u$ is the option value from node upper node at n+1.
- $V_d$ is the option value from the lower node at n+1.
- r is the risk-free interest rate.
- $\Delta t$ is the step size between time slices of the model.

See the code below for an implementation of the Cox Ross Rubinstein Model.

**Example 5.1**

Consider pricing a European Call option with the following parameters, X = \$60, $S_0$ = \$50, r = 5%, $\sigma = 0.2$, $\Delta t = 0.01$, N = 100.

Using the following MATLAB implementation for the Binomial Cox Ross Rubinstein Model we have:

```
>> oPrice=binPriceCRR(60,50,0.05,0.2,0.01,100,'CALL',false)

oPrice= 1.628
```

## 6. Analysis

Combining the results above we create the tables below, where the parameters for the European call option are X = \$60, $S_0$ = \$50, r = 5%, σ = 0.2, and T = 1. To use the Binomial Cox Ross Rubinstein method we add the parameters Δt = 0.01, N = 100.

Note: We shall compare the option prices of the four models concerning the Black Scholes value of a call option.

Consider the following table:

**Table 6.1**

| Stock Price | Black-Scholes value of Equation | Implicit Method | Explicit Method | Crank-Nicolson Method | Binomial CRR Method |
|---|---|---|---|---|---|
| \$50 | \$1.6237 | \$1.5826 | \$1.6209 | \$1.6216 | \$1.6279 |
| \$60 | \$6.2704 | \$5.6352 | \$6.259 | \$6.2641 | \$6.2584 |
| \$70 | \$13.9353 | \$10.1133 | \$13.7997 | \$13.8731 | \$13.9346 |
| \$80 | \$23.1795 | \$10.7415 | \$22.5200 | \$22.7579 | \$23.1783 |
| \$90 | \$32.9821 | \$6.8721 | \$30.8951 | \$31.3900 | \$32.9807 |
| \$100 | \$42.9375 | \$38.0492 | \$38.0492 | \$38.0492 | \$42.9372 |

Note: The Black-Scholes value of the call option was calculated using the built-in MATLAB code [Call, Put] = blsprice(Price, Strike, Rate, Time, Volatility).

Table 6.1 shows that for different stock prices the Explicit, the Crank-Nicolson, and the Binomial CRR method generally agree with the Black-Scholes value of an option whereas the Implicit Method did not generate similar values.

Furthermore, we can look at the convergence of the four models concerning the Black Scholes value of the option.

For the values of M and L where M is the number of intervals up to the maximum stock price and L is the number of intervals up until the expiry date where L=2M we shall calculate using the different models the price of a call option above with stock price $50 and strike price $60 relative to the Black Scholes value of $1.6237.

**Table**

| M | L | Implicit Method | Explicit Method | Crank-Nicolson Method | Binomial CRR Method |
|---|---|---|---|---|---|
| 10 | 20 | $1.3236 | $1.4413 | $1.4791 | $1.5710 |
| 20 | 40 | $1.5046 | $1.5655 | $1.5741 | $1.6104 |
| 30 | 60 | $1.5487 | $1.5967 | $1.6011 | $1.6310 |
| 40 | 80 | $1.5652 | $1.6080 | $1.6110 | $1.6136 |
| 50 | 100 | $1.5732 | $1.6133 | $1.6156 | $1.6279 |

From the above table 6.2, we can see that as M and L get larger the accuracy of the Crank Nicolson Finite Difference Method compared to the Binomial CRR method gradually increases. That is, the Crank-Nicolson method converges to the actual option price faster than the Binomial CRR model. Therefore we conclude that the Crank-Nicolson method of Finite Differencing is more stable than the Binomial CRR Model.

# Conclusion

Options are financial instruments that are useful in the purchasing of stocks. There are many ways to price these options. Three of these ways were assessed in detail and one method was briefly spoken about. Out of the four methods that were introduced, through my analysis, I discovered that only three of these methods returned consistent values: namely the Explicit method, the Crank-Nicolson method, and the Binomial CRR method. Furthermore, of these three methods, it was discovered that the Crank-Nicolson method is more appropriate in the pricing of options when the number of intervals for both time and the stock price increases.

## **References**

Causon, D. M., & Mingham, C. G. (2010). *Introductory finite difference methods for PDEs*. Bookboon. Retrieved from http://www.cs.man.ac.uk/~fumie/tmp/introductory-finite-difference-methods-for-pdes.pdf

Consulting, G. (n.d.). Option Pricing - Finite Difference Methods. Retrieved April 18, 2019, from http://goddardconsulting.ca/option-pricing-finite-diff-index.html

Emmanuel, F. S., Temitayo, O. J., & Oluwaseyi, A. A. On the Robustness of Binomial Model and Finite Difference Method for Pricing European Options. Retrieved April 20, 2019, from https://www.scribd.com/document/230138816/On-the-Robustness-of-Binomial-Model-and-Finite-Difference-Method-for-Pricing-European-Options

Higham, D. J. (2004). *An introduction to financial option valuation: mathematics, stochastics, and computation* (Vol. 13). Cambridge University Press.

Hull, J. C. (2012). Options, Futures, And Other Derivatives (Eight Edition). *New Jersey: Prentice-Hall*.

Teall, J. L. (2018). *Financial trading and investing*. Academic Press.

# Appendix

Implicit Finite Difference Method Code:

**MATLAB Function: finDiffImplicit**

```
function oPrice = finDiffImplicit(X,S0,r,sig,Svec,tvec,oType)
% Function to calculate the price of a vanilla European
% Put or Call option using the implicit finite difference method
%
% oPrice = finDiffImplicit(X,r,sig,Svec,tvec,oType)
%
% Inputs: X - strike
%       : S0 - stock price
%: r - risk-free interest rate
%: sig - volatility
%: Svec - Vector of stock prices (i.e. grid points)
%: tvec - Vector of times (i.e. grid points)
%       : oType - must be 'PUT' or 'CALL'.
%
% Output: oPrice - the option price
%
% Notes: This code focuses on details of the implementation of the
%        implicit finite difference scheme.
%        It does not contain any programmatic essentials such as error
```

```
%       checking.
%       It does not allow for optional/default input arguments.
%       It is not optimized for memory efficiency, speed, or
%       use of sparse matrices.

% Author: Phil Goddard (phil@goddardconsulting.ca)
% Date  : Q4, 2007

% Get the number of grid points
M = length(Svec)-1;
N = length(tvec)-1;
disp(M)
disp(N)
% Get the grid sizes (assuming equispaced points)
dt = tvec(2)-tvec(1);

% Calculate the coefficients
% To do this we need a vector of j points
j = 0:M;
sig2 = sig*sig;
aj = (dt*j/2).*(r - sig2*j);
bj = 1 + dt*(sig2*(j.^2) + r);
cj = -(dt*j/2).*(r + sig2*j);
```

```
% Pre-allocate the output
price(1:M+1,1:N+1) = nan;

% Specify the boundary conditions
switch oType
    case 'CALL'
        % Specify the expiry time boundary condition
        price(:,end) = max(Svec-X,0);
        % Put in the minimum and maximum price boundary conditions
        % assuming that the largest value in the Svec is
        % chosen so that the following is true for all time
        price(1,:) = 0;
        price(end,:) = (Svec(end)-X)*exp(-r*tvec(end:-1:1));
    case 'PUT'
        % Specify the expiry time boundary condition
        price(:,end) = max(X-Svec,0);
        % Put in the minimum and maximum price boundary conditions
        % assuming that the largest value in the Svec is
        % chosen so that the following is true for all time
        price(1,:) = (X-Svec(end))*exp(-r*tvec(end:-1:1));
        price(end,:) = 0;
end
```

```
% Form the tridiagonal matrix
B = diag(aj(3:M),-1) + diag(bj(2:M)) + diag(cj(2:M-1),1);
[L,U] = lu(B);

% Solve at each node
offset = zeros(size(B,2),1);
for idx = N:-1:1
    offset(1) = aj(2)*price(1,idx);
    % offset(end) = c(end)*price(end,idx); % This will always be zero
    price(2:M,idx) = U\(L\(price(2:M,idx+1) - offset));
end

% Calculate the option price
oPrice = interp1(Svec,price(:,1),S0);
```

Explicit Finite Difference Method Code:

**MATLAB Function: finDiffExplicit**

```
function oPrice = finDiffExplicit(X,S0,r,sig,Svec,tvec,oType)

% Function to calculate the price of a vanilla European

% Put or Call option using the explicit finite difference method

%

% oPrice = finDiffExplicit(X,r,sig,Svec,tvec,oType)

%

% Inputs: X - strike

%       : S0 - stock price

%: r - risk-free interest rate

%: sig - volatility

%: Svec - Vector of stock prices (i.e. grid points)

%: tvec - Vector of times (i.e. grid points)

%       : oType - must be 'PUT' or 'CALL'.

%

% Output: oPrice - the option price

%

% Notes: This code focuses on details of the implementation of the

%        explicit finite difference scheme.

%        It does not contain any programmatic essentials such as error

%        checking.
```

```
%       It does not allow for optional/default input arguments.

%       It is not optimized for memory efficiency, speed, or

%       use of sparse matrices.


% Author: Phil Goddard (phil@goddardconsulting.ca)

% Date  : Q4, 2007


% Get the number of grid points

M = length(Svec)-1;

N = length(tvec)-1;

% Get the grid sizes (assuming equispaced points)

dt = tvec(2)-tvec(1);


% Calculate the coefficients

% To do this we need a vector of j points

j = 1:M-1;

sig2 = sig*sig;

j2 = j.*j;

aj = 0.5*dt*(sig2*j2-r*j);

bj = 1-dt*(sig2*j2+r);

cj = 0.5*dt*(sig2*j2+r*j);


% Pre-allocate the output
```

```
price(1:M+1,1:N+1) = nan;


% Specify the boundary conditions
switch oType
    case 'CALL'
        % Specify the expiry time boundary condition
        price(:,end) = max(Svec-X,0);
        % Put in the minimum and maximum price boundary conditions
        % assuming that the largest value in the Svec is
        % chosen so that the following is true for all time
        price(1,:) = 0;
        price(end,:) = (Svec(end)-X)*exp(-r*tvec(end:-1:1));
    case 'PUT'
        % Specify the expiry time boundary condition
        price(:,end) = max(X-Svec,0);
        % Put in the minimum and maximum price boundary conditions
        % assuming that the largest value in the Svec is
        % chosen so that the following is true for all time
        price(1,:) = (X-Svec(end))*exp(-r*tvec(end:-1:1));
        price(end,:) = 0;
end


% Form the tridiagonal matrix
```

```
A = diag(bj);  % Diagonal terms

A(2:M:end) = aj(2:end); % terms below the diagonal

A(M:M:end) = cj(1:end-1); % terms above the diagonal


% Calculate the price at all interior nodes

offsetConstants = [aj(1); cj(end)];

for i = N:-1:1

    price(2:end-1,i) = A*price(2:end-1,i+1);

    % Offset the first and last terms

    price([2 end-1],i) = price([2 end-1],i) + ...

        offsetConstants.*price([1 end],i+1);

end


% Calculate the option price

oPrice = interp1(Svec,price(:,1),S0);
```

**MATLAB Function: finDiffCN**

```
function oPrice = finDiffCN(X,S0,r,sig,Svec,tvec,oType)
% Function to calculate the price of a vanilla European
% Put or Call option using the Crank-Nicolson finite difference method
%
% oPrice = finDiffCN(X,r,sig,Svec,tvec,oType)
%
% Inputs: X - strike
%       : S0 - stock price
%: r - risk-free interest rate
%: sig - volatility
%: Svec - Vector of stock prices (i.e. grid points)
%: tvec - Vector of times (i.e. grid points)
%       : oType - must be 'PUT' or 'CALL'.
%
% Output: oPrice - the option price
%
% Notes: This code focuses on details of the implementation of the
%        Crank-Nicolson finite difference scheme.
%        It does not contain any programmatic essentials such as error
%        checking.
%        It does not allow for optional/default input arguments.
%        It is not optimized for memory efficiency, speed, or
```

```
%        use of sparse matrices.

% Author: Phil Goddard (phil@goddardconsulting.ca)
% Date  : Q4, 2007

% Get the number of grid points
M = length(Svec)-1;
N = length(tvec)-1;
% Get the grid sizes (assuming equispaced points)
dt = tvec(2)-tvec(1);

% Calculate the coefficients
% To do this we need a vector of j points
j = 0:M;
sig2 = sig*sig;
aj = (dt/4)*(sig2*(j.^2) - r*j);
bj = -(dt/2)*(sig2*(j.^2) + r);
cj = (dt/4)*(sig2*(j.^2) + r*j);

% Pre-allocate the output
price(1:M+1,1:N+1) = nan;

% Specify the boundary conditions
```

```
switch oType
    case 'CALL'
        % Specify the expiry time boundary condition
        price(:,end) = max(Svec-X,0);
        % Put in the minimum and maximum price boundary conditions
        % assuming that the largest value in the Svec is
        % chosen so that the following is true for all time
        price(1,:) = 0;
        price(end,:) = (Svec(end)-X)*exp(-r*tvec(end:-1:1));
    case 'PUT'
        % Specify the expiry time boundary condition
        price(:,end) = max(X-Svec,0);
        % Put in the minimum and maximum price boundary conditions
        % assuming that the largest value in the Svec is
        % chosen so that the following is true for all time
        price(1,:) = (X-Svec(1))*exp(-r*tvec(end:-1:1));
        price(end,:) = 0;
end

% Form the tridiagonal matrix
C = -diag(aj(3:M),-1) + diag(1-bj(2:M)) - diag(cj(2:M-1),1);
[L,U] = lu(C);
D = diag(aj(3:M),-1) + diag(1+bj(2:M)) + diag(cj(2:M-1),1);
```

```
% Solve at each node
offset = zeros(size(D,2),1);
for idx = N:-1:1
    if length(offset)==1
        offset = aj(2)*(price(1,idx)+price(1,idx+1)) + ...
            cj(end)*(price(end,idx)+price(end,idx+1));
    else
        offset(1) = aj(2)*(price(1,idx)+price(1,idx+1));
        offset(end) = cj(end)*(price(end,idx)+price(end,idx+1));
    end
    price(2:M,idx) = U\(L\(D*price(2:M,idx+1) + offset));
end

% Calculate the option price
oPrice = interp1(Svec,price(:,1),S0);
```

Binomial Pricing Model Code:

```
function oPrice = binPriceCRR(X,S0,r,sig,dt,steps,oType,earlyExercise)
% Function to calculate the price of a vanilla European or American
% Put or Call option using a Cox Ross Rubinstein binomial tree.
%
% Inputs: X - strike
%       : S0 - stock price
%: r - risk-free interest rate
%: sig - volatility
%: dt – the size of time steps
%: steps - number of time steps to calculate
%       : oType - must be 'PUT' or 'CALL'.
%       : earlyExercise - true for American, false for European.
%
% Output: oPrice - the option price
%
% Notes: This code focuses on details of the implementation of the Cox Ross Rubinstein (CRR)
%        algorithm.
%        It does not contain any programmatic essentials such as error
%        checking.
%        It does not allow for optional/default input arguments.
%        It is not optimized for memory efficiency or speed.
```

```
% Author: Phil Goddard (phil@goddardconsulting.ca)
% Date  : Q4, 2007

% Calculate the Cox-Ross Rubinstein model parameters
a = exp(r*dt);
u = exp(sig*sqrt(dt));
d = 1/u;
p = (a-d)/(u-d);

% Loop over each node and calculate the Cox Ross Rubinstein underlying price tree
priceTree = nan(steps+1,steps+1);
priceTree(1,1) = S0;
for idx = 2:steps+1
    priceTree(1:idx-1,idx) = priceTree(1:idx-1,idx-1)*u;
    priceTree(idx,idx) = priceTree(idx-1,idx-1)*d;
end

% Calculate the value at expiry
valueTree = nan(size(priceTree));
switch oType
    case 'PUT'
        valueTree(:,end) = max(X-priceTree(:,end),0);
    case 'CALL'
```

```
        valueTree(:,end) = max(priceTree(:,end)-X,0);
end

% Loop backward to get values at the earlier times
steps = size(priceTree,2)-1;
for idx = steps:-1:1
    valueTree(1:idx,idx) = ...
        exp(-r*dt)*(p*valueTree(1:idx,idx+1) ...
        + (1-p)*valueTree(2:idx+1,idx+1));
    if earlyExercise
        switch oType
            case 'PUT'
                valueTree(1:idx,idx) = ...
                    max(X-priceTree(1:idx,idx),valueTree(1:idx,idx));
            case 'CALL'
                valueTree(1:idx,idx) = ...
                    max(priceTree(1:idx,idx)-X,valueTree(1:idx,idx));
        end
    end
end

% Output the option price
oPrice = valueTree(1);
```