UNIVERSITY OF WATERLOO

ACTSC 971

FINANCE 2

Technical Report on A Study of Numerical Solution on Black-Scholes Model

Name:
Yifei Deng

13 April, 2020

1 About the Paper

The main concern of the paper [1] is the numerical solution of the Black-Scholes model [2] proposed by Black, Scholes, and Merton for a simple European-type option using different finite difference algorithms. The Black-Scholes model is discussed and an explicit finite difference algorithm is implemented for the numerical approximation to the model. The semi-implicit algorithm discussed in [3] is also introduced for comparison with the explicit scheme. In the analysis of the algorithm for this paper [1], stability condition of the explicit method is established, the accuracy of the explicit method is tested by estimation of the relative error in L_1 norm and a comparison of numerical outcomes with the values obtained by Semi-implicit scheme is given.

2 Abstract

This report mainly focuses on reproducing most of the work that has been done in the paper [1] and introducing additional finite difference algorithms in solving the problem of pricing a European-type option. After a brief introduction to the Black-Scholes equation, the financial and mathematical assumptions of the model are discussed in the report. Numerical methods, namely the finite difference methods are employed to solve the European-type options. In terms of the finite domain discretization in stock price and time, four algorithms, namely, the explicit, semi-implicit, fully implicit and Crank-Nicolson algorithm are fully developed and a comparison of these algorithms is performed based on some illustrative examples. The numerical results later in the report showed that these finite difference algorithms are capable of solving the European option price with fixed boundaries. In the report, I will also discuss the effectiveness of the algorithms in the aspect of computational cost, accuracy, and convergence and give further recommendations upon the numerical results.

Throughout the whole report, for simplicity, all numerical results are based on pricing a European-type option. The main reason is the exact solution of the European-type option is known, then it is clear to compare between the algorithms and the exact solution. Base on that, we can thereby make appropriate recommendations and further applications (e.g. pricing an American type of option with moving boundary) in pricing the option in the financial market.

3 Methodology

The purpose of the report is to understand how the numerical methods can be applied to price European type of option and to examine how well the numerical methods performed in terms of pricing the European type of option. In a real financial market, investors have exposures to different kinds of risks and therefore, have different risk preferences. However, in the model, one can not account for all the risk factors encountered in the financial market. Consequently, negligence of risk is an essential assumption when it comes to making the valuation of the portfolio independent of each investor's risk preference. Under such a setting, the expected value of the portfolio is equal to the risk-free interest rate. Thus, the only risk present in the model is reflected by the risk-neutral probability, from which it is just an abstract assemble of risk factors existing in the financial market I.e.a risk measure. The essential idea of option pricing lies in the approximation to the value changes in the Black-Scholes model under this risk-neutral measure. To price a European type of option, in-depth methods in finite difference methods are implemented. After a discretization of the domain in stock price and time horizon, Taylor's series is applied on each grid point to approximate each partial derivative in the Black-Scholes equations using linear difference approximations. This in terms renders a system of linear equations include all the grid point both in time and price. To solve this system, the explicit, semi-implicit algorithms (introduced by the author), fully implicit method and Crank-Nicolson method (as part of my work) are the main focus in this report. To solve the explicit scheme, a direct iterated backward solution is given; to solve semi-implicit, implicit and Crank-Nicolson, the LU decomposition is employed. Further concerns investigated for the algorithms are stability, convergence, computational cost and the accuracy of the algorithm regardless of the running error.

4 Introduction

An option is a financial derivative that gives its holder the right to sell or buy a specific amount of a particular asset (e.g. stock) at a specific price, called the strike price and a specific date called the maturity date. There are two simplest types of options that have been traded in the financial market for a long time: options that can be only exercised at the maturity date are called the European type of options, while options that can be exercised anytime before or upon the maturity date are called the American type of options. An option that gives the right for its holder to buy the underlying assets is known as a call option, while an option that conferring the right to sell the underlying assets is known as a put option.

Nowadays, since thousands of options are traded for each day in the financial market, how to accurately and correctly pricing an option becomes a crucial topic. And thanks to the crowning achievement of Black, Scholes ,and Merton to derive the Balck-Scholes's partial differential equations [2], a second-order partial differential equation with respect to the time horizon t and the underlying asset price S,

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

one can price an option exactly using a closed-form equation derived from (1). However, in many practical situations, there will be no closed-form solution, like the complex exotic option. Therefore, the implementation of a numerical method will be one of the solutions for this kind of problem. As the main concern of this report, one type of numerical methods: finite difference methods will be discussed and investigated. Nonetheless, we will need to first state the assumptions of the underlying model (1) and some notations before introducing the finite difference methods to price a European-type option.

5 Basic Setting of Black-Scholes Model

Here are some financial and mathematical assumptions I will use throughout the whole report along with some simple notations:

- Returns on underlying stock price (S) follow a random walk and are of log-normal distributed
- The volatility (σ) of the stock price and the risk-free interest rate (r) are known and constant across the time (t) horizon
- The stock market is assumed to be liquid and perfect and one can assume continuous trading or delta hedging without the consideration of tax, dividend and transaction cost
- V(S,t) is the general option price when the distinction is needed, we use C(S,t) to denote the call option price and P(S,t) to denote the put option price; T is the time at expiry and K is the exercise price

- No arbitrage opportunities exist in the market
- All assets are divisible and the market is efficient

Given the assumptions from above, the finite difference method can be introduced in the following section.

6 Finite Difference Method

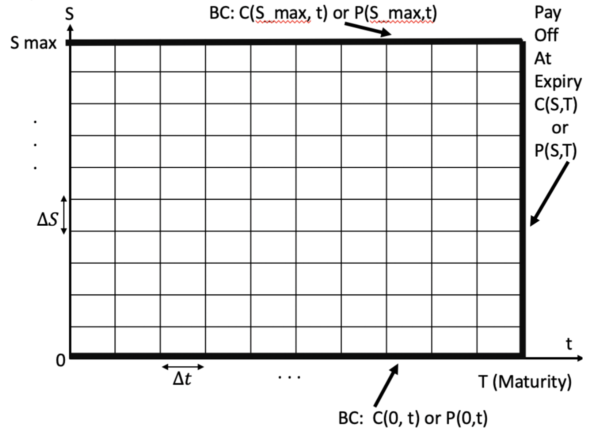
The finite difference method is a form of numerical analysis to solve certain partial derivative equations using linear difference approximations in the iterative scheme. Hence, in this case, this method is used to solve the Black-Scholes equation (1) for the fair price of a European type of option.

6.1 Domain Discretisation

Firstly, we will need to begin by discretizing both the stock price (S) and time (t) domain:

According to Figure 1., for time domain, we divided interval [0, T] into N equally sized sub-interval with size Δt, where n = 0,. . .,N, i.e. N = number of time steps; for the stock price; We then divided interval [0, S_{max}] into M equally sized sub-interval with size ΔS, where i = 0,. . .,M, i.e. M = number of asset steps. The total grid points are, therefore, $(N + 1)\times(M + 1)$.

Figure 1: Illustration of Domain Discretisation

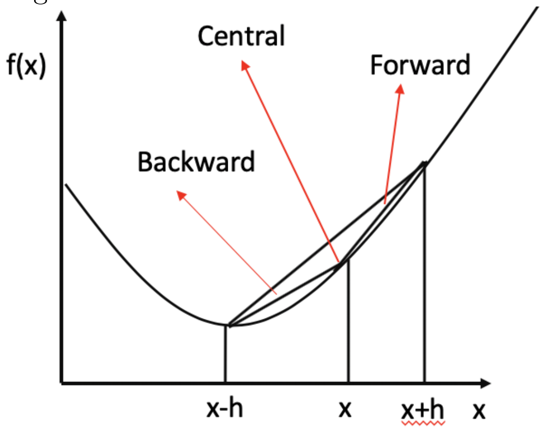

S
BC: C(S_max, t) or P(S_max,t)
S max
Pay Off At Expiry C(S,T) or P(S,T)
ΔS
0
Δt
t
T (Maturity)
BC: C(0, t) or P(0,t)

Noted that the price of the stock can be unbounded, so according to the paper [1], it is necessary to introduce an artificial boundary S_{max}. Normally, the choice of S_{max} will be 3-4 times the strike price. The interval from 0 to S_{max} is the area of interest. Knowing that truncating the price domain by indicating S_{max} to be the price boundary will lead to truncated error. However, for the sake of computational conveniences, it is fine to have it. It is also worthy of notice that there will be three boundary conditions, one at the expiry of the option and the other two are on the boundary of the stock price. They will be discussed in the later section. We will move on to the next step of the finite difference method: discretize the Black-Scholes partial differential equations (1).

6.2 PDE Discretisation

In this section, consider the function of one variable f(x) shown in Figure 2., by using the forward and backward scheme of the Taylor series around point x, the point we would like to estimate for f(x), the following approximations can be derived with the order of accuracy:

Figure 2: Illustration of PDE Discretisation



- Forward approximation of 1st derivative of f(x)

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

- Backward approximation of 1st derivative of f(x)

$$f'(x) = \frac{\mathrm{f}(x) - \mathrm{f}(x-h)}{\mathrm{h}} + O(h) \tag{3}$$

- Central approximation of 1st derivative of f(x)

$$f'(x) = \frac{\mathrm{f}(x+h) - \mathrm{f}(x-h)}{2h} + O(h^2) \tag{4}$$

- Standard approximation of 2nd derivative of f(x)

$$f''(x) = \frac{\mathrm{f}(x+h) - 2f(x) + \mathrm{f}(x-h)}{\mathrm{h}^2} + O(h^2) \tag{5}$$

Since the iterated scheme of finite difference method is depending on the boundary conditions of the problem. So it is also crucial to determine the boundary conditions for a European call/put option.

6.3 Boundary Conditions for European Call/Put Option

First, denote C_i^n call option price at time n and asset price i. And then denote P_i^n put option price at time n and asset price i. Thus, the boundary conditions based on put-call parity are derived as follow based on the put-call parity:

For European call option (i = 0,...,M, n = 1,...,N):

$$\begin{cases} C_i^T & = \max(S_T - K, 0), @T \\ C_0^n & = 0, S \to 0 \\ C_{S_t}^n & = S_t - K\mathrm{e}^{-r(T-n\Delta t)}, S_t \to \infty \end{cases} \tag{6}$$

For European put option (i = 0,...,M, n = 1,...,N):

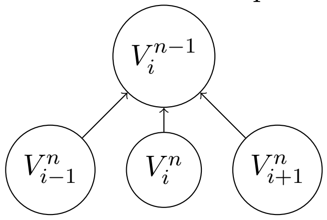
$$\begin{cases} P_i^T & = \max(K - S_T, 0), @T \\ P_0^n & = K\mathrm{e}^{-r(T-n\Delta t)}, S \to 0 \\ P_{S_t}^n & = 0, S_t \to \infty \end{cases} \tag{7}$$

Now, I have set up all the conditions for implementing the finite difference method, so it is time to introduce different finite difference algorithms.

6.4 Explicit Finite Difference Algorithm

Firstly, denote the numerical value of option price as V_i^n, where price steps i = 0,...,M, time steps n = 0,...,N

Figure 3: Illustration of Explicit Scheme



$$\begin{tikzcd} & V_i^{n-1} & \\ V_{i-1}^n \arrow[ur] & V_i^n \arrow[u] & V_{i+1}^n \arrow[ul] \end{tikzcd}$$

For the explicit algorithm, as Figure 3. shown, backward difference formula in time of $\frac{\partial V}{\partial t}$(3), central difference approximation of $\frac{\partial V}{\partial S}$ (4) and standard approximation of $\frac{\partial^2 V}{\partial S^2}$ (5) are applied to each component of the Black-Scholes equations (1). Hence the Black-Scholes equations become (in this case, the order of accuracy of the explicit algorithm is $O(\Delta t, \Delta S^2)$),

$$\frac{V_i^n - V_i^{n-1}}{\Delta t} + ri\Delta S \frac{V_{i+1}^n - V_{i-1}^n}{2\Delta S} + \frac{1}{2}\sigma^2 (i\Delta S)^2 \frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{\Delta S^2} = rV_i^n \quad (8)$$

Simplify and rearrange (8) to find coefficient for V_{i-1}^n, V_i^n and V_{i+1}^n, this yield that (8) is now,

$$a_i V_{i-1}^n + b_i V_i^n + c_i V_{i+1}^n = V_i^{n-1} \quad (9)$$

where i = 1,...,M, n = N-1,...,0 and

$$\begin{aligned} a_i &= -\frac{r\Delta t i}{2} + \frac{1}{2}\sigma^2 i^2 \Delta t, \\ b_i &= 1 - \sigma^2 i^2 \Delta t - r\Delta t, \\ c_i &= \frac{r\Delta t i}{2} + \frac{1}{2}\sigma^2 i^2 \Delta t. \end{aligned}$$

Thus, the equation (9) will be the explicit backward finite difference algorithm and observed that the equation can be solved backwardly from time node n to time node n-1 iteratively due to the three known boundaries in section 6.3.

The explicit algorithm is the most straight forward algorithm to employ among the four algorithms, but it might generate unstable option value if the grid is not fine enough for accurate interpolation. Therefore, the choice of S_{max}, Δt ,and ΔS are crucial for the stability of the explicit algorithm. To obtain the constraint for a finer grid, a comprehensive analysis of the stability of the explicit method is conducted below as part of my work.

Stability Analysis of Explicit Algorithm (Von-Neumann)

For the explicit algorithm, the truncated error originated from numerical approximation might grow over time if the algorithm is unstable. It is natural to take a look at the truncated error and impose constraint to ensure that the error term does not expand exponentially overtime for the option price. Since in the real market, we don't want our option price to be negatively or positively large and hugely jumped around. Let U_i^n be the actual solution to the equation (1) on the grid point($i\Delta S$,$n\Delta t$) and V_i^n be the numerical approximation using the explicit scheme to the exact solution on the same grid point. While the truncated error is defined as $T_i^n = U_i^n - V_i^n = e_i^n$, since V_i^n follows the finite explicit equation (8), then so is the error term e_i^n, this yield that,

$$\frac{e_i^n - e_i^{n-1}}{\Delta t} + ri\Delta S \frac{e_{i+1}^n - e_{i-1}^n}{2\Delta S} + \frac{1}{2}\sigma^2 (i\Delta S)^2 \frac{e_{i+1}^n - 2e_i^n + e_{i-1}^n}{\Delta S^2} = re_i^n \quad (10)$$

It is assumed that the error oscillating over time and the starting point of the error is hard to find, but the behavior of the error term can be considered as part of the Fourier Series at each price node from [4], derived as follow:

$$e_i = \hat{e} exp(\mathbf{i}\theta i) \quad (11)$$

where $\mathbf{i} = \sqrt{-1}$, k = $\frac{2\pi}{S_{max}}$ the wave length of each price node and $\theta = k\Delta S$

After simplifying and rearranging (10), I combined the fact (11) at each node with Euler equation, and substitute them back into (10), (10) will become,

$$S(k) = \frac{\hat{e}^n}{\hat{e}^{n-1}} = (1 - \sigma^2 i^2 \Delta t - r\Delta t + r\mathbf{i}\Delta tisin(\theta) + \sigma^2 i^2 \Delta tcos(\theta)) \quad (12)$$

Observed that, to achieve stability, we must have $|S(k)| \leq 1$, then it is equivalent to show:

- On the real part: $|1 - \sigma^2 i^2 \Delta t - r\Delta t| \leq 1, \forall i$, I use the fact that $\Delta t = \frac{T}{N}$, $S_{max} = M\Delta S$ and I choose i = M $\rightarrow$ When S_{max} gets large enough, it is not hard to see $0 < \frac{\Delta t}{\Delta S^2} \leq \frac{1}{\sigma^2 S_{max}^2}$
- On the imaginary part: $r\mathbf{i}\Delta tisin(\theta) + \sigma^2 i^2 \Delta tcos(\theta) = \sigma^2 i^2 \Delta t[cos(\theta) + \frac{ri\Delta t}{\sigma^2 i^2 \Delta t}\mathbf{i}sin(\theta)]$, where the imaginary part behaves like a damped oscillator contributing to the source of error and to make the error term

decay away, we must have $\sigma^2 i^2 \Delta t \leq 1$, using the same facts as the real part $\rightarrow$ It is also not hard to see $0 < \frac{\Delta t}{\Delta S^2} \leq \frac{1}{\sigma^2 S_{max}^2}$

Thus, the explicit scheme is conditionally stable for

$$0 < \frac{\Delta t}{\Delta S^2} \leq \frac{1}{\sigma^2 S_{max}^2} \tag{13}$$

And a stably finer grid can, therefore, be established by (13). Apart from the explicit method, the author in the paper [1] has also discussed the semi-implicit method to compare and show the merits of the explicit method. The semi-implicit method will be developed in the following section.

6.5 Semi-implicit Finite Difference Algorithm

For the semi-explicit algorithm, backward difference formula in time of $\frac{\partial V}{\partial t}$(3), forward difference approximation of $\frac{\partial V}{\partial S}$ (2) and standard approximation of $\frac{\partial^2 V}{\partial S^2}$ (5) are applied to each component of the Black-Scholes equations (1). Hence the Black-Scholes equations become (in this case, the order of accuracy of the semi-implicit algorithm is $O(\Delta t, \Delta S)$,

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} + ri\Delta S \frac{V_{i+1}^{n+1} - V_i^{n+1}}{\Delta S} + \frac{1}{2}\sigma^2 (i\Delta S)^2 \frac{V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}}{\Delta S^2} = rV_i^{n+1} \tag{14}$$

Simplify and rearrange (14) to find coefficient for V_{i-1}^{n+1}, V_i^{n+1} and V_{i+1}^{n+1}, this yield that (14) is now,

$$a_i V_{i-1}^{n+1} + b_i V_i^{n+1} + c_i V_{i+1}^{n+1} = V_i^n \tag{15}$$

where,

$$\begin{aligned} a_i &= -\frac{1}{2}\sigma^2 i^2 \Delta t, \\ b_i &= 1 + r\Delta t + ri\Delta t + \sigma^2 i^2 \Delta t, \\ c_i &= -r\Delta t i - \frac{1}{2}\sigma^2 i^2 \Delta t. \end{aligned}$$

Consider rewriting the equation (15) into system, we will obtain the semi-implicit algorithm:

$$AV^{n+1} = b^n \tag{16}$$

where n = N-1,...0, A$\in R^{(M-1)\times(M-1)}$, $V^{n+1} \in R^{M-1}$, $b^n \in R^{M-1}$

We then can transform the system (16) into matrix form, note that the terms $a_1 V_0^{n+1}$ and $c_{M-1} V_{M-1}^{n+1}$ are known from the boundary conditions,

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{M-2} & b_{M-2} & c_{M-2} \\ & & & & a_{M-1} & b_{M-1} \end{pmatrix} \begin{pmatrix} V_1^{n+1} \\ V_2^{n+1} \\ V_3^{n+1} \\ \vdots \\ V_{M-2}^{n+1} \\ V_{M-1}^{n+1} \end{pmatrix} = \begin{pmatrix} V_1^{n} \\ V_2^{n} \\ V_3^{n} \\ \vdots \\ V_{M-2}^{n} \\ V_{M-1}^{n} \end{pmatrix} + \begin{pmatrix} -a_1 V_0^{n+1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -c_{M-1} V_{M-1}^{n+1} \end{pmatrix}.$$

It is easy to observe that matrix A, in this case, is strictly diagonally dominant, then A is non-singular. As a result, for saving computational cost, I can use the LU decomposition to solve this linear system (16) to obtain option value at each time step and price step.

The semi-implicit algorithm is unconditional stable in this case. According to standard matrix algebra, an equation of the form (16), specifically for a matrix like A in (16), stability is achieved if and only if: $\|A\|_\infty \leq 1$, where $\|A\|_\infty$ is the infinity norm. Therefore, we can validate this statement by checking if the infinite norm of matrix A is smaller or equal to 1, which is trivial to show. As for the next step, I have extended the set of algorithms to Fully implicit and Crank-Nicolson algorithms. Both algorithms are fully derived and discussed in the following sections.

6.6 Fully-implicit Finite Difference Algorithm

For the Fully-implicit algorithm, as Figure 4.shown, forward difference formula in time of $\frac{\partial V}{\partial t}$(2), central difference approximation of $\frac{\partial V}{\partial S}$ (4) and standard approximation of $\frac{\partial^2 V}{\partial S^2}$ (5) are applied to each component of the Black-Scholes equations (1). Hence the Black-Scholes equations become (in this case, the order of accuracy of the Fully-implicit algorithm is $O(\Delta t, \Delta S^2)$,

$$\frac{V_i^{n+1} - V_i^n}{\Delta t} + ri\Delta S \frac{V_{i+1}^n - V_{i-1}^n}{2\Delta S} + \frac{1}{2}\sigma^2 (i\Delta S)^2 \frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{\Delta S^2} = rV_i^n \tag{17}$$

Simplify and rearrange (17) to find coefficient for V_{i-1}^n, V_i^n and V_{i+1}^n, this yield that (17) is now,

$$a_i V_{i-1}^n + b_i V_i^n + c_i V_{i+1}^n = V_i^{n+1} \tag{18}$$

Figure 4: Illustration of Fully-implicit Scheme

$$\begin{tikzcd} & V_i^{n+1} & \\ V_{i+1}^n \arrow[ur] & V_i^n \arrow[u] & V_{i-1}^n \arrow[ul] \end{tikzcd}$$

where,

$$\begin{aligned} a_i &= \frac{1}{2}r\Delta t i - \frac{1}{2}\sigma^2 i^2 \Delta t, \\ b_i &= 1 + \sigma^2 i^2 \Delta t + r\Delta t, \\ c_i &= -\frac{1}{2}r\Delta t i - \frac{1}{2}\sigma^2 i^2 \Delta t. \end{aligned}$$

Consider rewriting the equation (18) into system, we will obtain fully-implicit algorithm:

$$AV^{n+1} = b^n \tag{19}$$

where n = N-1,...0, A$\in R^{(M-1)X(M-1)}$, $V^{n+1} \in R^{M-1}$, $b^n \in R^{M-1}$

We then can transform the system (18) into matrix form, note that the terms $a_1 V_0^{n+1}$ and $c_{M-1}V_{M-1}^{n+1}$ are known from the boundary conditions,

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{M-2} & b_{M-2} & c_{M-2} \\ & & & & a_{M-1} & b_{M-1} \end{pmatrix} \begin{pmatrix} V_1^{n+1} \\ V_2^{n+1} \\ V_3^{n+1} \\ \vdots \\ V_{M-2}^{n+1} \\ V_{M-1}^{n+1} \end{pmatrix} = \begin{pmatrix} V_1^n \\ V_2^n \\ V_3^n \\ \vdots \\ V_{M-2}^n \\ V_{M-1}^n \end{pmatrix} + \begin{pmatrix} -a_1 V_0^{n+1} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -c_{M-1}V_{M-1}^{n+1} \end{pmatrix}.$$

Again, it is easy to observe that matrix A in this case is strictly diagonally dominant, then A is non-singular. As a result, for saving computational cost, I can also use the LU decomposition to solve this linear system (19) to obtain option value at each time step and price step.

The fully-implicit algorithm is again unconditional stable in this case and we can validate this statement using the same rationale as we discussed in the semi-implicit scheme part by checking the infinite norm of matrix A is smaller or equals to 1, which is again trivial to show. Next, we will develop our last algorithm in this report: Crank-Nicolson finite difference algorithm.

6.7 Crank-Nicolson Finite Difference Algorithm

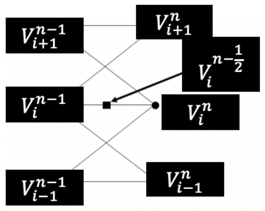
For the Crank-Nicolson algorithm, as Figure 5. shown, in this case, the method is like an average of both fully implicit and explicit method. Thus, I will need 6 nodes to price the option value at center node $n - \frac{1}{2}$. Hence the Black-Scholes equations become (in this case, the order of accuracy of the Fully-implicit algorithm is $O(\Delta t^2, \Delta S^2)$,

$$-\overline{a_i}V_{i-1}^{n-1} + (1 - \overline{b_i})V_i^{n-1} - \overline{c_i}V_{i+1}^{n-1} = \overline{a_i}V_{i-1}^{n} + (1 + \overline{b_i})V_i^{n} + \overline{c_i}V_{i+1}^{n} \quad (20)$$

where,

$$\begin{aligned}\overline{a_i} &= \frac{\Delta t}{4}(\sigma^2 i^2 - ri),\\ \overline{b_i} &= -\frac{\Delta t}{2}(\sigma^2 i^2 + r),\\ \overline{c_i} &= \frac{\Delta t}{4}(\sigma^2 i^2 + ri).\end{aligned}$$

Figure 5: Illustration of Crank-Nicolson Algorithm



Consider rewriting the equation (20) into system, we will obtain Crank-Nicolson algorithm:

$$AV^{n-1} = BV^n + z^{n-1} + z^n \quad (21)$$

where n = N-1,...0, A,B$\in R^{(M-1)\times(M-1)}$, $V^n, V^{n-1} \in R^{M-1}$, $z^n, z^{n-1} \in R^{M-1}$

$$A = \overline{a_1}\begin{bmatrix} 1-\bar{b}_1 & -\bar{c}_1 & 0 & \cdots & 0 & 0 \\ -\bar{a}_2 & 1-\bar{b}_2 & -\bar{c}_2 & \cdots & 0 & 0 \\ & -\bar{a}_3 & 1-\bar{b}_3 & \cdots & 0 & 0 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & \cdots & -\bar{a}_{M-1} & 1-\bar{b}_{M-1} \end{bmatrix}\overline{c_{M-1}}; \quad V^n = \begin{pmatrix} V_1^n \\ V_2^n \\ \vdots \\ V_{M-2}^n \\ V_{M-1}^n \end{pmatrix}; \quad z^{n-1} = \begin{pmatrix} \overline{a_1}V_0^{n-1} \\ 0 \\ \vdots \\ 0 \\ \overline{c_{M-1}}V_{M-1}^{n-1} \end{pmatrix}; \quad z^n = \begin{pmatrix} \overline{a_1}V_0^n \\ 0 \\ \vdots \\ 0 \\ \overline{c_{M-1}}V_{M-1}^n \end{pmatrix}$$

$$B = \overline{a_1}\begin{bmatrix} 1+\bar{b}_1 & \bar{c}_1 & 0 & \cdots & 0 & 0 \\ \bar{a}_2 & 1+\bar{b}_2 & \bar{c}_2 & \cdots & 0 & 0 \\ & \bar{a}_3 & 1+\bar{b}_3 & \cdots & 0 & 0 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 0 & \cdots & \bar{a}_{M-1} & 1+\bar{b}_{M-1} \end{bmatrix}\overline{c_{M-1}}$$

We then can transform the system (21) into matrix form, note that the terms $a_1V_0^n$, $c_{M-1}V_{M-1}^n$, $a_1V_0^{n-1}$ and $c_{M-1}V_{M-1}^{n-1}$ are known from the boundary conditions,

Again, it is easy to observe that matrix A in this case is strictly diagonally dominant, then A is non-singular. As a result, for saving computational cost, I can also use the LU decomposition to solve this linear system (21) to obtain option value at each time step and price step.

The Crank-Nicolson algorithm is unconditional stable in this case and we can validate this statement by performing Von-Neumann stability analysis the same as how we did in the explicit scheme part. One can obtain the symbol S(k) as follow by using change of variables, x $= log(S)$, $\tau = T - t$, and $|S(k)| < 1$, $\forall k$ [5]:

$$|\mathbf{S(k)}|^2 = \frac{\left[1-r-\frac{\Delta\tau\sigma^2}{2\Delta x^2}(1-\cos(\theta))\right]^2 + \left[\frac{\Delta\tau}{2\Delta x}\left(r-\frac{1}{2}\sigma^2\right)\sin(\theta)\right]^2}{\left[1+r-\frac{\Delta\tau\sigma^2}{2\Delta x^2}(1-\cos(\theta))\right]^2 + \left[\frac{\Delta\tau}{2\Delta x}\left(r-\frac{1}{2}\sigma^2\right)\sin(\theta)\right]^2}, \text{ where } \theta = k\Delta x$$

In general, I have derived and discusses all four algorithms, including fully implicit and Crank-Nicolson method by my own, from finite difference methods. A comprehensive analysis on stability for each algorithm has also been discussed in this section. Next, to validate the effectiveness and convergence of the algorithms and to make comparison among algorithms, I will need to proceed to the numerical results section to verify which algorithm is more practical to use for option pricing and which is not.

7 Numerical Results

In this section, I will focus on presenting the numerical results I reproduced from the paper [1] and performing analysis on convergence for all four algorithms in numerical approaches. Lastly, I will make an overall comparison among all algorithms in the aspect of computational cost and accuracy. Some potential limitations for each algorithm will also be investigated during the numerical analysis.

7.1 Analytic Solution for European Call/Put Option

Before implementing finite difference algorithms, it is necessary to mention the exact solution that will be used to compare among all the algorithms. The exact solution of the European call/put option problem is calculated using the following formulas:

$$C(S,t) = S\Phi(d_1) - K\mathrm{e}^{-r(T-t)}\Phi(d_2) \tag{22}$$

$$P(S,t) = K\mathrm{e}^{-r(T-t)}\Phi(-d_2) - S\Phi(-d_1) \tag{23}$$

where $\Phi(\cdot)$is the CDF for a standard normal random variable, d_1 and d_2 are given by

$$d_{1,2} = \frac{\log(S/K) + (r \pm \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$$

7.2 Replications of Numerical Results from Paper [1]

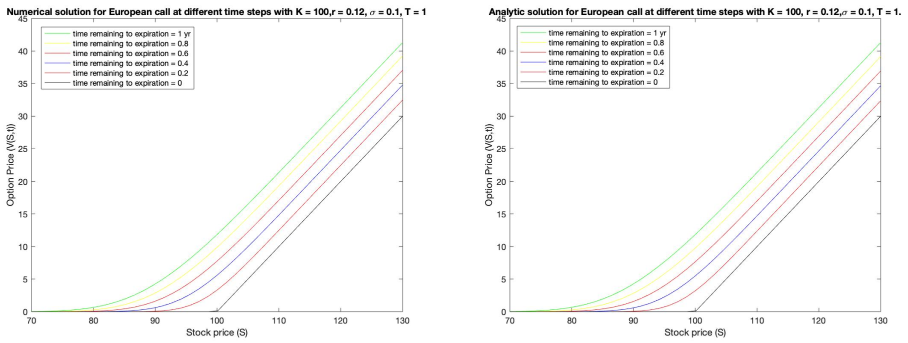


Figure 6: Comparison Between Exact Solution and Explicit Solution

Figure 6. from above is a part of the author's work for comparing the exact solution and the explicit solution on pricing European call option with

T = 1, r =0.12, K = 100, σ = 0.1, Δ t = 0.01, ΔS = 1.50, S_{max} = 130. On the one hand, I have successfully reproduced the same graph as in the paper [1]. On the other hand, comparing the one produced by using the explicit method with the exact solution, we can't observe any difference by naked eyes. Hence, we can say the explicit numerical scheme has a very small error to approximate the exact solution. According to Figure 7., by overlaying the analytic solution with the explicit scheme on pricing European Call option using the same parameters as Figure 6., one can also reach the same conclusion as previously stated.

Figure 7: Overlaying Plot Between Exact Solution and Explicit Solution

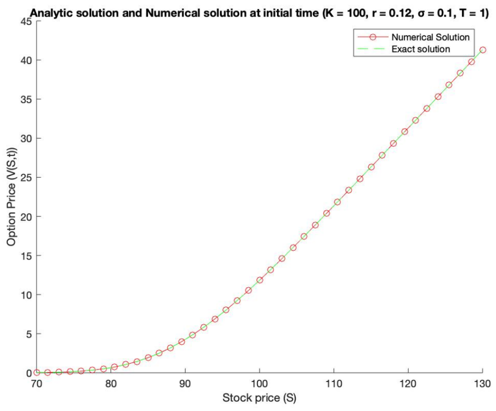


Figure 8: Relative Error for Explicit Scheme on Pricing European Call

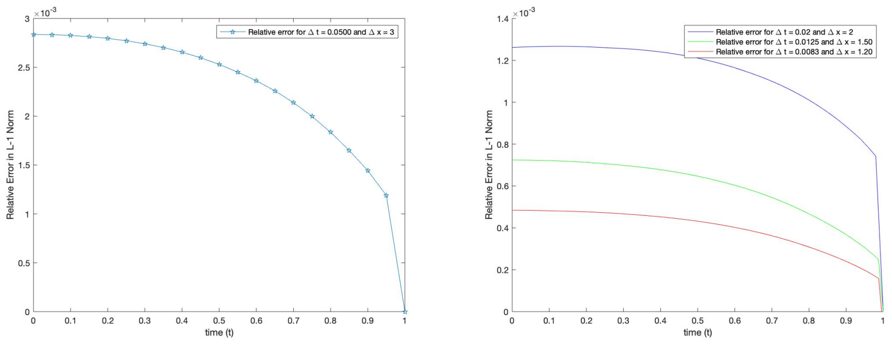
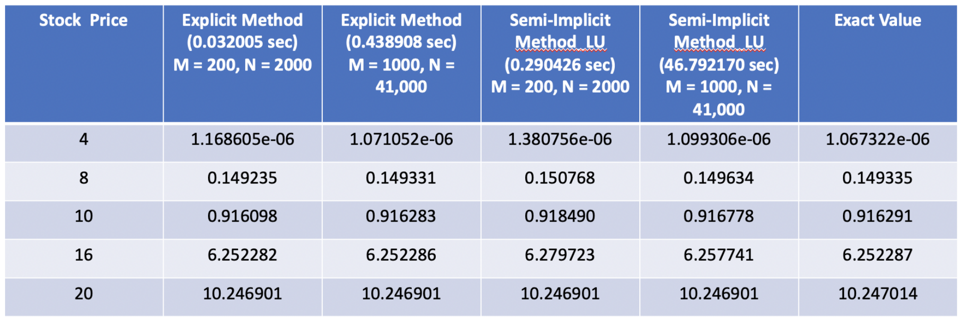


Figure 8. from above is a part of the author's work for relative Error of explicit Scheme on pricing European Call option with T = 1, r =0.12, K = 100, σ= 0.10, S_{max} = 130. For the convergence of explicit algorithm,

as we can see from the left plot of Figure 8. with other parameters holding constant, the relative error become smaller and smaller as time approaches to expiry, and the convergence of error is of order 10^{-3} which is somewhat acceptable. The right plot of Figure 8. is the relative error of explicit difference scheme for different temporal and spatial grid size, as we decrease the size of Δt and ΔS, i.e. increasing the number of iterations over the price and time domain (M and N), the relative error is decreasing, thereby it proves the convergence of the explicit difference scheme only if the grid we choose satisfy the condition derived in (13). (in this case we chose a finer grid for price and time from (13)). Again, the numerical results and conclusions are consistent with the author's work.



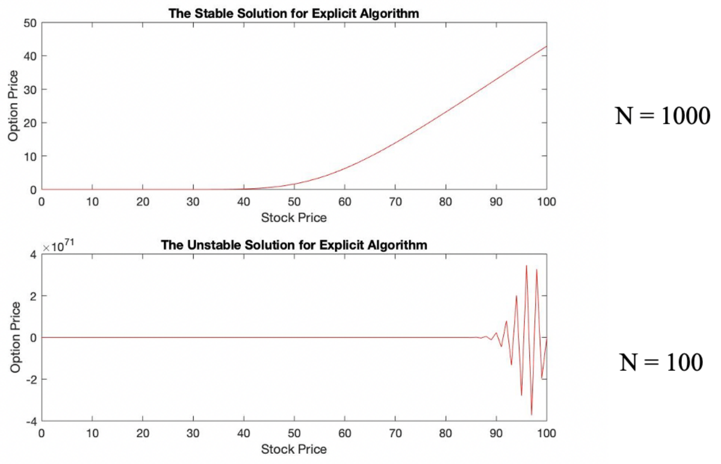
Stock Price	Explicit Method (0.032005 sec) M = 200, N = 2000	Explicit Method (0.438908 sec) M = 1000, N = 41,000	Semi-Implicit Method_LU (0.290426 sec) M = 200, N = 2000	Semi-Implicit Method_LU (46.792170 sec) M = 1000, N = 41,000	Exact Value
4	1.168605e-06	1.071052e-06	1.380756e-06	1.099306e-06	1.067322e-06
8	0.149235	0.149331	0.150768	0.149634	0.149335
10	0.916098	0.916283	0.918490	0.916778	0.916291
16	6.252282	6.252286	6.279723	6.257741	6.252287
20	10.246901	10.246901	10.246901	10.246901	10.247014

Table 1. Comparison between the semi-implicit method and explicit method for pricing European call option
S_max = 20, T = 0.25, K = 10, r = 0.1, σ = 0.4

According to the last numerical results from paper [1], the comparison has been made between the semi-implicit and the explicit method of pricing a European call option. According to the reproduced Table 1., all the numerical results are consistent with the author's work for explicit method and it also shows that as the number of time and price steps (M and N) increase with other parameters holding unchanged, both the explicit and semi-implicit method is getting closer to the exact solution. It is seen that the explicit algorithm will give better results than the semi-implicit algorithm due to the lower order of accuracy for the semi-implicit method. In this case, the results and conclusions I made are consistent with the paper [1], the only part different from the paper is the solution for the semi-implicit method. The result from Table 1. about the semi-implicit method is slightly different from the results in the paper [1] solution. According to my analysis, the reason for the difference is possibly due to the different parameters used like S_{max} or boundary conditions imposed from [3].

So, what happened if the grid we choose is unstable for the explicit algorithm? As an illustration for this problem, I use the following parameters K = 60, S_{max} = 100, r = 0.05, T = 1, σ = 0.2, M = 100 to give the plots in Figure 9. The only difference is for the lower plot, it uses time step N = 100 that violated the stable condition (13) and for the upper plot, it uses a finer time step N = 1000 satisfied the stable condition (13). We can clearly observe that, as the condition (13) being violated, the option price will expand exponentially both on the positive and negative directions across the stock price. When we are pricing options in the real financial market, we certainly do not want to observe this odd phenomenon accord to the lower plot in Figure 9. This teaches us that we need to be cautious about the choice of S_{max}, M and N when using the explicit algorithm to price an option.

Figure 9: Illustration for the statement that choice of the size for time steps, price steps and largest stock price matters for Euro Call



Generally, in this section, I have reproduced most of the work done by the author in paper [1] along with detailed observations and discussion of the numerical results. I have also indicated the limitation when implementing the explicit method for a certain choice of parameters. In the next subsection, I will proceed to perform convergence analysis on the other three algorithms.

7.3 Numerical Convergence Analysis

To validate the numerical solutions of the other three finite difference algorithms approach to the exact solution as both grid interval and time step sizes are reduced, three convergence analysis has been conducted in this section on corresponding algorithms.

7.3.1 Semi-implicit Algorithm

In this case, in terms of using relative error, I implement the percentage error, which is essentially the same as the relative error but with a better visualization. The parameters that holding unchanged are T = 1, r = 0.12, K = 100, σ= 0.10. According to Figure 10., the error is decreasing from the left plot to the right plot as we change the original large Δt and ΔS to smaller discretization parameters Δt and ΔS, which shows the convergence of the semi-implicit difference scheme.

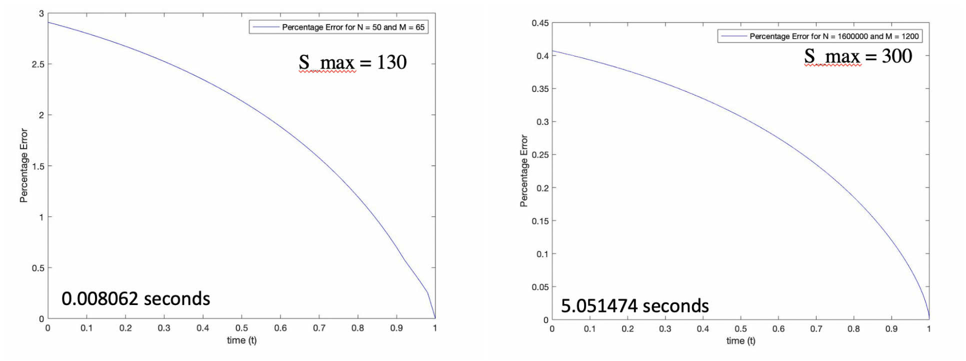


Figure 10: Convergence Analysis for Semi-implicit Algorithm

7.3.2 Fully-implicit Algorithm

In this case, the same as the previous sub-section, I implement the percentage error, which is essentially the same as the relative error but with a better visualization. The parameters holding unchanged are T = 1, r = 0.12, K = 100, σ= 0.10. According to Figure 11., the error is decreasing from the left plot to the right plot as we change the original large Δt and ΔS to smaller discretization parameters Δt and ΔS, which shows the convergence of the Fully-implicit difference scheme.

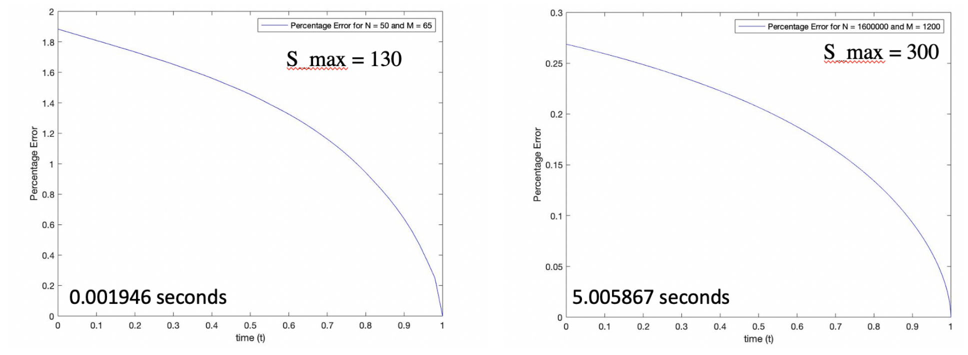


Figure 11: Convergence Analysis for Fully-implicit Algorithm

7.3.3 Crank-Nicolson Algorithm

In this case, the same as the previous sub-section, I implement the percentage error, which is essentially the same as the relative error but with a better visualization. The parameters holding unchanged are T = 1, r = 0.12, K = 100, σ= 0.10. According to Figure 12., the error is decreasing from the left plot to the right plot as we change the original large Δt and ΔS to smaller discretization parameters Δt and ΔS, which shows the convergence of the Crank-Nicolson difference scheme.

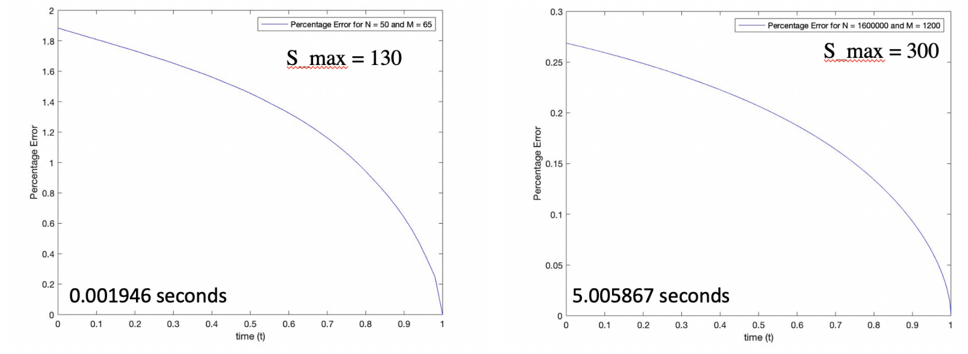
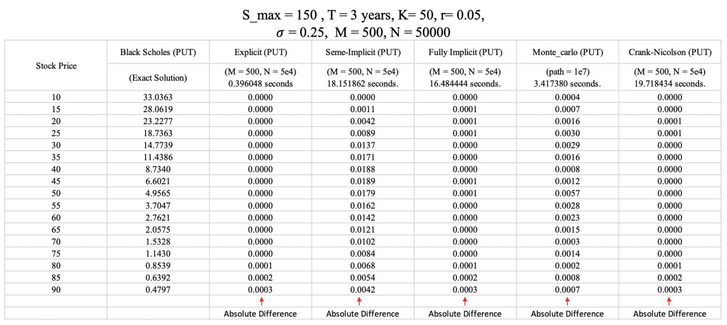


Figure 12: Convergence Analysis for Crank-Nicolson Algorithm

As a summary of the sub-section, all algorithms in this sub-section are converging to the exact solution. The limitations of these three algorithms are also quite similar, as we look at the left plots on both Figure 10., Figure 11. and Figure 12., the order of error is higher with smaller time step(N) and price step (M). So, to obtain higher accuracy of the price of the option, we need to have a large number for time steps as we look at the right plots on both Figure 10., Figure 11. and Figure 12. At the same time, by dividing finitely more pieces over the domain, one has to increase the execution time as shown in both Figure 10., Figure 11. and Figure 12. A possible reason for Crank-Nicolson to adapt this phenomenon could be the fact that although the Crank-Nicolson scheme is of second-order accurate both in time and space, i.e. $O(^2, \Delta t^2)$, in order to be used, the scheme should be applied with a prohibitively small time step size (i.e. small Δt).

7.4 Overall Comparison

7.4.1 Accuracy



S_max = 150 , T = 3 years, K= 50, r= 0.05,
σ = 0.25, M = 500, N = 50000

Stock Price	Black Scholes (PUT)	Explicit (PUT)	Seme-Implicit (PUT)	Fully Implicit (PUT)	Monte_carlo (PUT)	Crank-Nicolson (PUT)
	(Exact Solution)	(M = 500, N = 5e4) 0.396048 seconds	(M = 500, N = 5e4) 18.151862 seconds.	(M = 500, N = 5e4) 16.484444 seconds.	(path = 1e7) 3.417380 seconds.	(M = 500, N = 5e4) 19.718434 seconds.
10	33.0363	0.0000	0.0000	0.0000	0.0004	0.0000
15	28.0619	0.0000	0.0011	0.0001	0.0007	0.0000
20	23.2277	0.0000	0.0042	0.0001	0.0016	0.0001
25	18.7363	0.0000	0.0089	0.0001	0.0030	0.0001
30	14.7739	0.0000	0.0137	0.0000	0.0029	0.0000
35	11.4386	0.0000	0.0171	0.0000	0.0016	0.0000
40	8.7340	0.0000	0.0188	0.0000	0.0008	0.0000
45	6.6021	0.0000	0.0189	0.0001	0.0012	0.0000
50	4.9565	0.0000	0.0179	0.0001	0.0057	0.0000
55	3.7047	0.0000	0.0162	0.0000	0.0028	0.0000
60	2.7621	0.0000	0.0142	0.0000	0.0023	0.0000
65	2.0575	0.0000	0.0121	0.0000	0.0015	0.0000
70	1.5328	0.0000	0.0102	0.0000	0.0003	0.0000
75	1.1430	0.0000	0.0084	0.0000	0.0014	0.0000
80	0.8539	0.0001	0.0068	0.0001	0.0002	0.0001
85	0.6392	0.0002	0.0054	0.0002	0.0008	0.0002
90	0.4797	0.0003	0.0042	0.0003	0.0007	0.0003
		↑	↑	↑	↑	↑
		Absolute Difference	Absolute Difference	Absolute Difference	Absolute Difference	Absolute Difference

Table 2. Accuracy comparison among all algorithms for European put option, where Column 2 is the exact solution

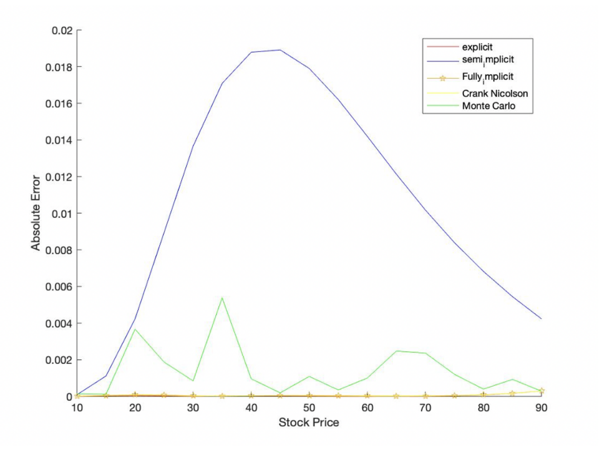


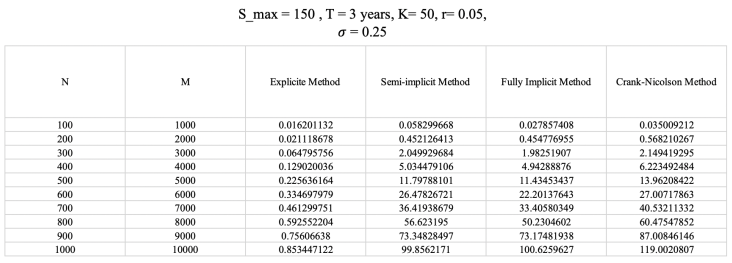
Figure 13: Absolute Error for all Algorithm for a European put option

In this sub-section, we will take a look at Table 2. about the accuracy among all algorithms on pricing a European put option, where the second column of the table is the exact solution for the European put option. The remaining columns are the absolute difference between the exact value of the put option and the numerical results from each algorithm.

Observed from Table 2. that both explicit, fully implicit and Crank-Nicolson algorithms perform well on pricing accurately the European put option. However, for the computational cost, the explicit algorithm is more efficient than the other two algorithms. As for the semi-implicit algorithm, it is both lower in accuracy and higher in computational cost. Comparing all the finite difference algorithms with the classical approach for pricing the European put option using Monte-Carlo simulation discuss in [6], we could see the merits of finite difference method to have a more accurate and stable solution for pricing an option. We can also plot the accuracy table into Figure 13. to visualize the difference in term of accuracy on pricing the European option. Clearly, from the figure, both explicit, fully implicit and Crank-Nicolson are better than Monte-Carlo and Semi-implicit methods on pricing the European put option.

7.4.2 Computational Cost

In this sub-section, we will take a look at the execution time among all algorithms for pricing a European call option. According to Table 3., when the number of time steps(N) and price steps(M) increase, the execution time will increase accordingly, equivalently it will lead to a higher computational cost. Among the algorithms, Crank-Nicolson has the highest computational cost, while the explicit method has the lowest computational cost.



S_max = 150 , T = 3 years, K= 50, r= 0.05,
σ = 0.25

N	M	Explicite Method	Semi-implicit Method	Fully Implicit Method	Crank-Nicolson Method
100	1000	0.016201132	0.058299668	0.027857408	0.035009212
200	2000	0.021118678	0.452126413	0.454776955	0.568210267
300	3000	0.064795756	2.049929684	1.98251907	2.149419295
400	4000	0.129020036	5.034479106	4.94288876	6.223492484
500	5000	0.225636164	11.79788101	11.43453437	13.96208422
600	6000	0.334697979	26.47826721	22.20137643	27.00717863
700	7000	0.461299751	36.41938679	33.40580349	40.53211332
800	8000	0.592552204	56.623195	50.2304602	60.47547852
900	9000	0.75606638	73.34828497	73.17481938	87.00846146
1000	10000	0.853447122	99.8562171	100.6259627	119.0020807

Table 3. Execution time (in seconds) among all algorithm for a European call option

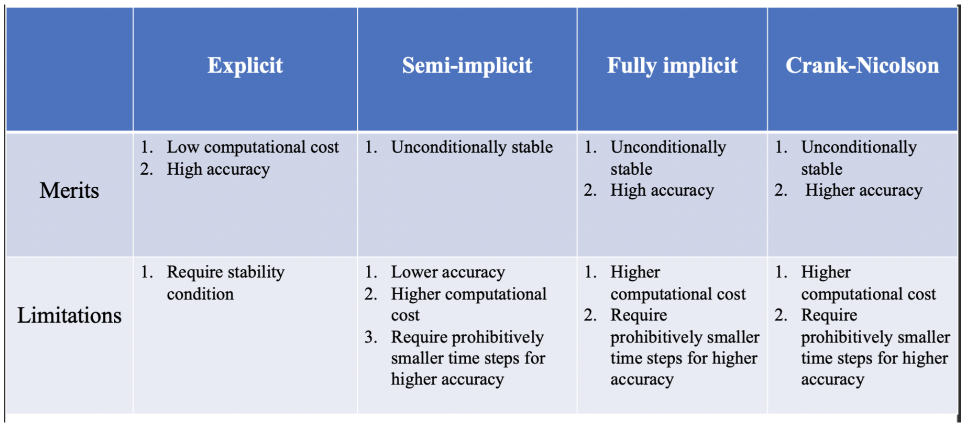
8 Conclusion

Generally, a total of four finite difference algorithms: explicit, semi-implicit, fully-implicit and Crank-Nicolson are introduced, derived and investigated in this report. Apart from reproducing most of the works in paper [1], this report also contains a detailed discussion of stability analysis for each algorithm and a series of numerical analyses on the convergence of the numerical solution from all algorithms to the exact solutions. Furthermore, numerical results for comparison on accuracy and computational cost among all algorithms when pricing a European type of option is also presented in the report. The numerical results and both the stability and convergence analysis showed that these finite difference algorithms are capable of solving the European option price with fixed boundaries in an effective way.

8.1 Recommendations

However, for future applications on complex exotic options like American option and basket option, we will need to select the one that is comparatively efficient among the four algorithms meaning that it will cost us less time and render higher accuracy when pricing an option in the financial market. Hence, a table is summarised as below for the convenience of comparison between each algorithm. According to the table below, since semi-implicit algorithm is both lower in accuracy and higher in computational cost, then it is the most undesirable one among the four. Secondly, since the fully-implicit method shows a similar computational cost with lower accuracy for the option price compares to the Crank-Nicolson algorithm according to sub-section 7.4.2., then Crank-Nicolson is more desirable. However, it is also true that the explicit finite difference algorithm also has an accurate approximation to the exact solution with lower computational cost, but has a constraint on the stability condition.

Thus, explicit and Crank-Nicolson finite difference algorithms are the favorable two among the four. As for recommendations, if one is more concerned about computational cost than accuracy, he or she should choose explicit finite difference method; If one is more concerned about the accuracy than the computational cost, then he or she should choose Crank-Nicolson finite difference method. It is worthy of notice that one has to be cautious of using both of the methods. As for the explicit method, we will need to pay additional care on choosing the grid points that satisfy the stability condition (13). As for Crank-Nicolson, we simply need to choose as small as both the Δt and ΔS as possible with the scarification of computational cost.



	Explicit	Semi-implicit	Fully implicit	Crank-Nicolson
Merits	1. Low computational cost 2. High accuracy	1. Unconditionally stable	1. Unconditionally stable 2. High accuracy	1. Unconditionally stable 2. Higher accuracy
Limitations	1. Require stability condition	1. Lower accuracy 2. Higher computational cost 3. Require prohibitively smaller time steps for higher accuracy	1. Higher computational cost 2. Require prohibitively smaller time steps for higher accuracy	1. Higher computational cost 2. Require prohibitively smaller time steps for higher accuracy

There are lots of further applications for the finite difference method algorithm such as pricing exotic types of options like American option with moving boundary and basket option. Even outside of the world of derivatives, the finite difference method can also be applied on actuarial science field on products like guarantees and insurance contracts

8.2 Problems and Further Improvements

The problems for the finite difference method are inherent since the underlying model for this numerical algorithm is the Black-Scholes equation [2], from which it assumes no-arbitrage, constant risk-free rate and volatility and log-normal distributed for stock returns. Apparently, in a real financial market, such assumptions will not hold. Therefore, further improvements for the setting of the algorithm can simply assume a non-constant risk-free rate and volatility or assume a different market scenario in either an active market or emerging market, for instance, implementation of sub-diffusive geometric Brownian motion [7] for the stock price in an emerging market with a lower volume of transactions. Additional improvements can also be made by extending this one factor Black-Scholes model to a multi-factor model, where extra factors like 2 or 3 more assets can be considered in the model. This will facilitate the pricing process while pricing options like basket option.

References

[1] Nurul Anwar, M. and Sazzad Andallah, L. (2018) A Study on Numerical Solution of Black-Scholes Model.Journal of Mathematical Finance,8, 372-381. doi:10.4236/jmf.2018.82024.

[2] F. Black, M.S. Scholes, The pricing of options and corporate liabilities, J. Political Econ. 81 (1973) 637–654.

[3] Dura, G. and Mosneagu, A.-M. (2010) Numerical Approximation of Black-Scholes Equation. An. Stiint. Univ."Al. I. Cuza" Iasi. Mat, 56, 39-64.

[4] Sterck, Hans De, and Paul Ullrich. NUMERICAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS. (2007.)

[5] Forsyth, P. A., Vetzal, K. R. (2017).Numerical computation for financial modeling.

[6] Richardson, M. (2009) Numerical Methods of Option Pricing, Thesis (PhD), University of Oxford

[7] Magdziarz, Marcin. (2009). Black-Scholes Formula in Subdiffusive Regime. Journal of Statistical Physics. 136. 553-564. 10.1007/s10955-009-9791-4.