MÄLARDALEN UNIVERSITY
SWEDEN

Solving Black-Scholes PDE by Crank-Nicolson and Hopscotch methods

Instructor: Jan Röman

Course: Analytical Finance I

Group 7:

Li Weitian

Hong Xi

Mälardale University
Division of Applied Mathematics
School of Education, Culture and Communication

Abstract

Our work is to use the Hopscotch and the Crank-Nicolson methods to solve European option prices, and we analyze the pricing results from these two methods by comparing to the pricing result generates from the Black-Scholes model. The accomplishment of our work is based on MATLAB applications.

Table of content

Introduction

Pricing options is crucial items for most of the financial institutes. Black-Scholes model is a well known tools for pricing different kinds of options. However, the model always leads to a direct calculation of the price which could make a high computational effort.

Hopscotch method and Crank-Nicolson method has a different nature. They approximate the option price with less computation efforts while ensuring a reasonable accuracy for the users. Application of these methods in option pricing could make the financial institute work more fast and efficient.

The applications of the Hopscotch method and Crank-Nicolson method for option pricing are the subject of this report. Our goal is not only to implement this method, but also to point out the differences of these approaches.

The report is organized as follows:
Section 1 is devoted to provide the knowledge of Black-Scholes PDE and the basic numerical approximation to derivatives.
In Section 2, we will apply the explicit scheme and fully implicit scheme to Black-Scholes PDE.
In Section 3, we introduce the Hopscotch method and Crank-Nicolson method which can be regard as a hybrid between the explicit and the fully implicit scheme.
Section 4 is due to the application of explicit scheme ,Hopscotch method and Crank-Nicolson method. We will illustrate the property and advantages of these methods by comparing the performance of them. The MATLAB build-in function: blsprice will be taken as a standard result for our comparison.

Conclusions will summarize all the achievement during this report.

1. Introduction of PDEs and Black-Sholes PDE

1.1 The black-Scholes Partial Differential Equation

Partial differential equations (PDEs) play a significant role in financial engineering. Due to the seminal work leading to the Black-Scholes equation, PDEs have become a major tool in **option valuation**.
We can use the Black-Scholes PDE to find the theoretical price $f\ (S,t)$ of a derivative security depending on the price S of one underlying asset at time t. This equation is derived from a stochastic differential equation that models the dynamics of underlying asset price and no arbitrage arguments:

$$\frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} + rS\frac{\partial f}{\partial S} - rf = 0, \tag{1}$$

Where,

r – risk-free interest rate and $\boldsymbol{\sigma}$ – asset price volatility

1.2 Numerical approximation of derivatives

The simple idea of approximating partial derivatives of a given PDE by finite differences is the fundamental sole for finite difference methods. As a tool for solving PDEs, this process transforms analytical differential equations into a set of algebraic equations.

As in many numerical algorithms, the starting point is a finite series approximation**.** Under suitable continuity and differentiability hypotheses, Taylor's theorem states that function $f(x)$ may be represented as:

$$f(x+h) = f(x) + \mathrm{h}f'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \cdots \tag{2}$$

Neglecting terms of h^2 and also the higher order, we will get:

The ***forward approximation*** for the derivative:

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

By similar reasoning, we can write:

$$f(x-h) = f(x) - \mathrm{h}f'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + \cdots \tag{4}$$

Rearranging terms, we obtain the ***backward approximation*** for the derivative:

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

However, more accurate approximation can be obtained by subtracting **(4)** from **(2)** and rearranging terms.

Doing this we obtain the ***central or symmetric approximation:***

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} + O(h^2) \tag{6}$$

Note: **(7)** does not imply that forward and backward approximations are disregarded. These differences are used to make efficient approximation depending on the type of boundary conditions.

1.3 Higher-order derivatives:

If we want to cope with the Black-Sholes equation, we have to approximate second-order derivatives. So we add **(2)** and **(4)***:*

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4), \tag{7}$$

After rearranging terms, we get:

$$f''(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + O(h^2), \tag{8}$$

Note: While solving PDE, the analytical function $\boldsymbol{f(S,t)}$ is transformed into a grid function of arguments $\boldsymbol{i\delta S}$ and $\boldsymbol{j\delta t}$, where $\boldsymbol{\delta S}$ and $\boldsymbol{\delta t}$ are discretization steps. In this framework, we will use the following notation:

$$f_{ij} = f(i\delta S, j\delta t) \tag{9}$$

We will deal with the discrete grid in later when applying Finite Difference Methods to Black-Sholes Equation.

1.4 Applying numerical approximation of derivatives to B-S PDE

We start solving **(1)** by setting up a discrete gird:

Suppose that $\boldsymbol{T}$ is the maturity time of an option, and $\boldsymbol{S_{max}}$ is a suitable large asset price, this $\boldsymbol{S_{max}}$ cannot be reached by $\boldsymbol{S(t)}$ within the time horizon we consider.

Note: Since the domain for the PDE is unbounded with respect to asset price, but we

must bound it in some way for computational purposes, so we need $\boldsymbol{S_{max}}$.
With respect to time and asset prices, the grid consists of points$(\boldsymbol{S}, \boldsymbol{t})$:

$$\begin{aligned} &\boldsymbol{S = 0, \delta S, 2\delta S, \ldots\ldots, M\delta S = S_{max}} \\ &\boldsymbol{t = 0, \delta t, 2\delta t, \ldots\ldots, N\delta t = T} \\ &\boldsymbol{f_{i,j} = f(i\delta S, j\delta t) \rightarrow the\ grid\ notation} \end{aligned} \tag{10}$$

After setting up the discrete grid **(10)**, we rewrite derivatives approximation using grid notation:

- ***Forward Difference:***

$$\frac{\partial f}{\partial S} = \frac{f_{i+1,j} - f_{i,j}}{\delta S}, \quad \frac{\partial f}{\partial t} = \frac{f_{i,j+1} - f_{i,j}}{\delta t} \tag{11 & 12}$$

- ***Backward Difference :***

$$\frac{\partial f}{\partial S} = \frac{f_{i,j} - f_{i-1,j}}{\delta S}, \quad \frac{\partial f}{\partial t} = \frac{f_{i,j} - f_{i,j-1}}{\delta t} \tag{13 &14}$$

- ***Central (or Symmetric) Difference:***

$$\frac{\partial f}{\partial S} = \frac{f_{i+1,j} - f_{i-1,j}}{2\delta S}, \quad \frac{\partial f}{\partial t} = \frac{f_{i,j+1} - f_{i,j-1}}{2\delta t} \tag{15 & 16}$$

- ***Second derivative:***

$$\frac{\partial^2 f}{\partial S^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\delta S^2} \tag{17}$$

1.5 Setting the boundary

For a put option with strike price K**,** the terminal condition (payoff) at expiration is:

$$f(S,T) = max\{K - S, 0\} \qquad \forall S \tag{18}$$

Let's consider a vanilla European put option. When $\boldsymbol{S}(\boldsymbol{t})$ is very large, the option is worthless, that is it will stay out-of-the-money

$$f(\boldsymbol{S_{max}}, t) = 0. \tag{19}$$

The value of $\boldsymbol{S_{max}}$ must be relatively large for this boundary condition to work properly.

When $\boldsymbol{S}(\boldsymbol{t})$ is zero, the asset price is zero. Therefore discounting back to time *t*, the payoff at expiration will be equal to:

$$f(0,t) = Ke^{-r(T-t)}. \tag{20}$$

In grid notation, boundary condition **(18) - (20)** has the following form:

$$\begin{aligned} \mathbf{f_{i,N} = max[K - i\delta S, 0]}, \quad & \mathbf{i = 0, 1, \dots, M} \\ \mathbf{f_{0,j} = Ke^{-r(N-j)\delta t}}, \quad & \mathbf{j = 0, 1, \dots, N} \\ \mathbf{f_{M,j} = 0}, \quad & \mathbf{j = 0, 1, \dots, N} \end{aligned} \tag{21}$$

2. An introduction of explicit scheme and fully implicit scheme

2.1 Explicit scheme

Considering a vanilla European **put option**, one of the possibilities is to approximate the derivatives in Black-Scholes PDE with respect to $\boldsymbol{S(t)}$ by a central difference and the derivative with respect to time by a backward difference:

$$\frac{f_{i,j} - f_{i,j-1}}{\delta t} + ri\delta S \frac{f_{i+1,j} - f_{i-1,j}}{2\delta S} + \frac{1}{2}\sigma^2 i^2 \delta S^2 \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\delta S^2} = rf_{i,j} \tag{22}$$

Since we already have a set of terminal conditions, see **(21)**, the equations must be solved backward in time.
Letting $j = N$, we have an unknown value, $f_{i,N-1}$, which is a function of three known values. This holds for each time layer when we solve this equation backward in time.

We get an **explicit scheme** after rewriting **(22)**:

$$f_{i,j-1} = a_i^* f_{i-1,j} + b_i^* f_{i,j} + c_i^* f_{i+1,j}$$

$$j = N-1, N-2, \dots, 1, 0; i = 1, 2, \dots, M-1 \tag{23}$$

Where

$$a_i^* = \frac{1}{2}\delta t(\sigma^2 i^2 - ri) \tag{24}$$

$$b_i^* = 1 - \delta t(\sigma^2 i^2 + r) \tag{25}$$

$$c_i^* = \frac{1}{2}\delta t(\sigma^2 i^2 + ri) \tag{26}$$

Note: In **(23)**, we have $f_{i,j-1}$ as an unknown value linked to three known values. This kind of schemes is called as explicit.

2.2 Fully implicit scheme

Now let’s price vanilla European **put option** again, but this time we approximate the Black-Scholes PDE by forward difference with respect to time and central difference with respect to asset prices. The difference equation in this case has the following form:

$$\frac{f_{i,j+1}-f_{i,j}}{\delta t} + ri\delta S\frac{f_{i+1,j}-f_{i-1,j}}{2\delta S} + \frac{1}{2}\sigma^2 i^2 \delta S^2 \frac{f_{i+1,j}-2f_{i,j}+f_{i-1,j}}{\delta S^2} = rf_{i,j} \qquad (27)$$

We can rewrite this equation as follows:

$$a_i f_{i-1,j} + b_i f_{i,j} + c_i f_{i+1,j} = f_{i,j+1}$$
$$i = 1,2,\dots,M-1;\ j = 0,1,\dots,N-1 \qquad (28)$$

For each **i,**

$$a_i = \frac{1}{2}ri\delta t - \frac{1}{2}\sigma^2 i^2 \delta t \qquad (29)$$

$$b_i = 1 + \sigma^2 i^2 \delta t + r\delta t \qquad (30)$$

$$c_i = -\frac{1}{2}ri\delta t - \frac{1}{2}\sigma^2 i^2 \delta t \qquad (31)$$

Note: In **(28)**, we have three unknown values linked to one known value $f_{i,j+1}$. This kind of schemes is called as fully implicit.

3. Hopscotch and Crank-Nicolson methods

3.1The Hopscotch method

When we solve a partial differential equation, we always create some kind of grid:

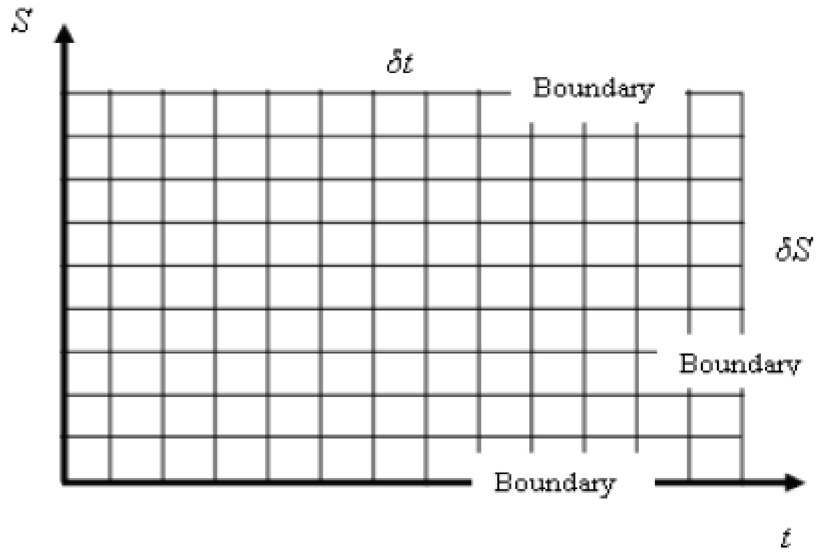

Figure 1

If we combine the forward- and backward differences and place the nodes as in the figure below:

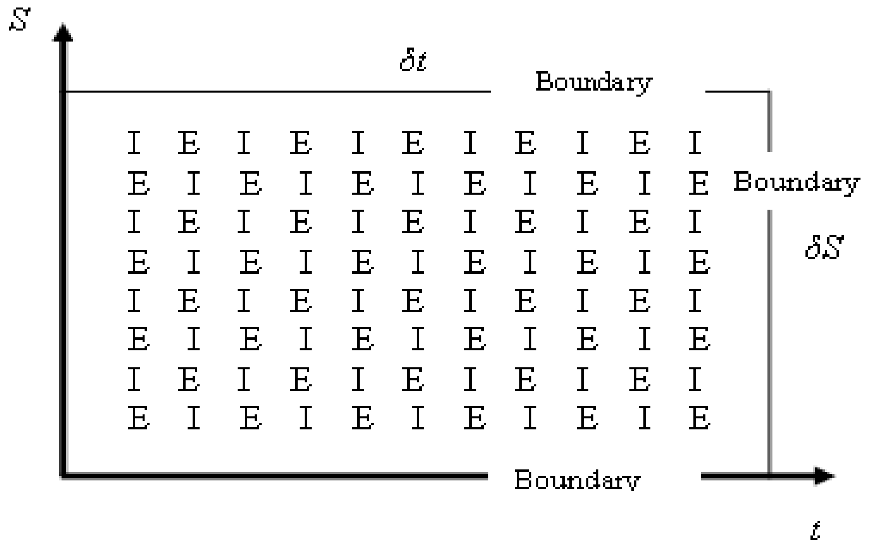

Figure 2

We alternate between the explicit and implicit calculations as we move from node to node. At each time, we first do all the calculations at the 'explicit nodes' in the usual way. We can then deal with the 'implicit nodes' without solving a set of simultaneous equations because the values at the adjacent nodes have already been calculated.

Moreover, by mix the nodes in this way, we can get almost the same accuracy as the Crank-Nicholson method. That is to say: the Hopscotch method, as well as the Crank-Nicholson method, can avoid the numerical instability, which is the disadvantage of the explicit scheme, and we will show this in section 4.

3.2 Crank-Nicolson method

Fully implicit method has attracting advantage -- *unconditional stability*, however, it could cause unnecessary computational complexity to price an option**.**
Yet another, a better way to price option is the Crank-Nicolson method (C-N), which may be regarded as a **hybrid** between the explicit and the fully implicit approach.
If we applying the C-N idea to the Black-Sholes equation, we figured out the following grid equation:

$$\frac{f_{ij}-f_{i,j-1}}{\delta t}+\frac{ri\delta S}{2}+\left(\frac{f_{i+1,j-1}-f_{i-1,j-1}}{2\delta S}\right)+\frac{ri\delta S}{2}\left(\frac{f_{i+1,j}-f_{i-1,j}}{2\delta S}\right)+ \frac{\sigma^2 i^2(\delta S)^2}{4}\left(\frac{f_{i+1,j-1}-2f_{i,j-1}+f_{i-1,j-1}}{(\delta S)^2}\right)+\frac{\sigma^2 i^2(\delta S)^2}{4}\left(\frac{f_{i+1,j}-2f_{i,j}+f_{i-1,j}}{(\delta S)^2}\right)= \frac{r}{2}f_{i,j-1}+\frac{r}{2}f_{ij}. \tag{32}$$

We rewrite **(33)** as:

$$-\alpha_i f_{i-1,j-1}+(1-\beta_i)f_{i,j-1}-\gamma_i f_{i+1,j-1}=\alpha_i f_{i-1,j}+(1+\beta_i)f_{ij}+\gamma_i f_{i+1,j} \tag{33}$$

Where:

$$\alpha_i=\frac{\delta t}{4}(\sigma^2 i^2-ri) \tag{34}$$

$$\beta_i=-\frac{\delta t}{2}(\sigma^2 i^2+r) \tag{35}$$

$$\gamma_i=\frac{\delta t}{4}(\sigma^2 i^2+ri) \tag{36}$$

4 MATLAB applications

Since all the formulas and schemes are given, we will apply Hopscotch method, Crank-Nicolson method into MATLAB to approximate the European option prices,

we will take vanilla European put option as our applications' example.

Note: Black-scholes model will be regarded as a **standard, or standard price** in all the comparing examples as followed.

4.1 Numerical instability of explicit scheme:

Let's use explicit scheme to approximate the put option value with different discrete asset prices, and compare its result with the Black-Sholes one.
There will be two curves: the red one stands for the **standard price** calculated by MATLAB's build-in function: **blsprice**, the blue one stands for our explicit scheme's price.

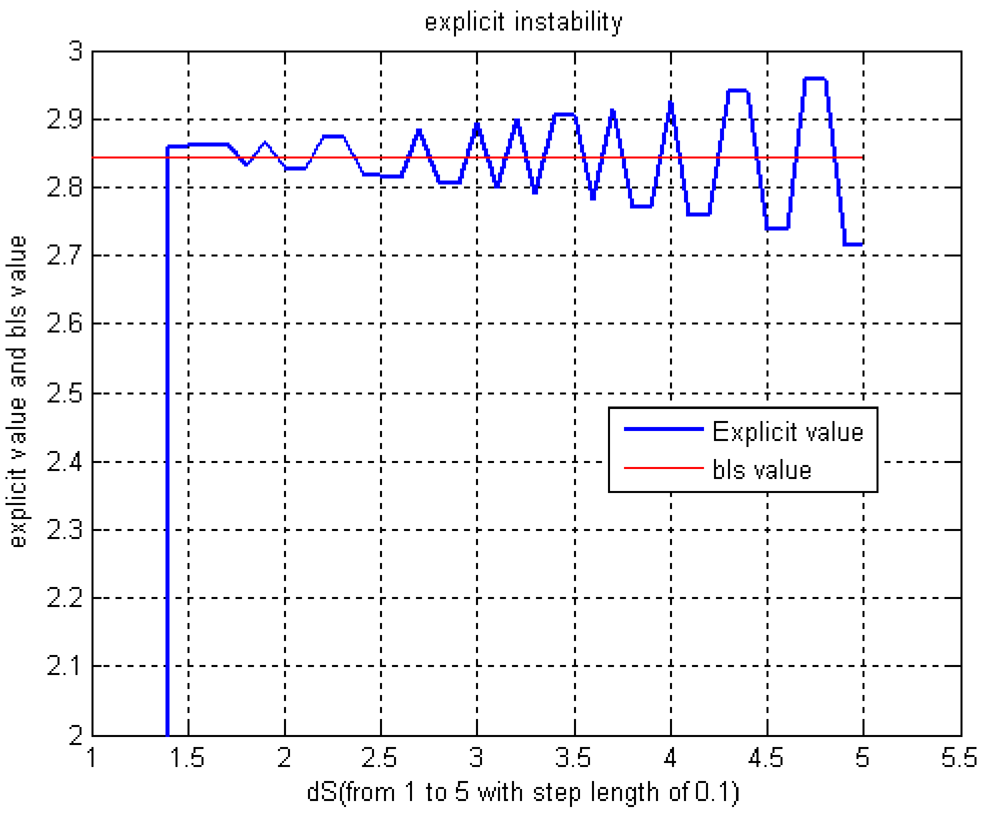

Figure 3

Analyzing this figure, we conclude that the accuracy of approximation increases as δS decreases. However, there exist some significant errors, even out of the bound, between $\delta S = 1$ and $\delta S = 1.5$.
Oscillations illustrated by this example are caused by numerical instability.

4.2 Hopscotch method vs. Crank-Nicolson method

As mentioned before, by mix the nodes of implicit and explicit scheme, the Hopscotch method can get almost the same result. Therefore, we would be very interested to see which one of these two is more **accurate** and **efficient** when pricing an option.

The following three figures are the results of pricing an European put Option by Black-Sholes model, Hopscotch method, and Crank-Nicolson method:

Suppose we set:
$\boldsymbol{S_0}(asset\ price) = 1$ to 99;
$\boldsymbol{K}\ (strike\ price) = 50$;
$\boldsymbol{r}\ \ (risk - free\ interest\ rate); = 0.1$;
$\boldsymbol{\sigma}\ (volatility) = 0.3$;
$\boldsymbol{T}\ (maturity) = 5/12$;
$\boldsymbol{dS}\ (discrete\ step\ of\ asset\ price) = 1$;
$\boldsymbol{dt}\ (discrete\ step\ of\ time) = 5/1200$;
$\boldsymbol{S_{max}}(boundary\ condition) = 100$;

And we get the figure for each one as followed:

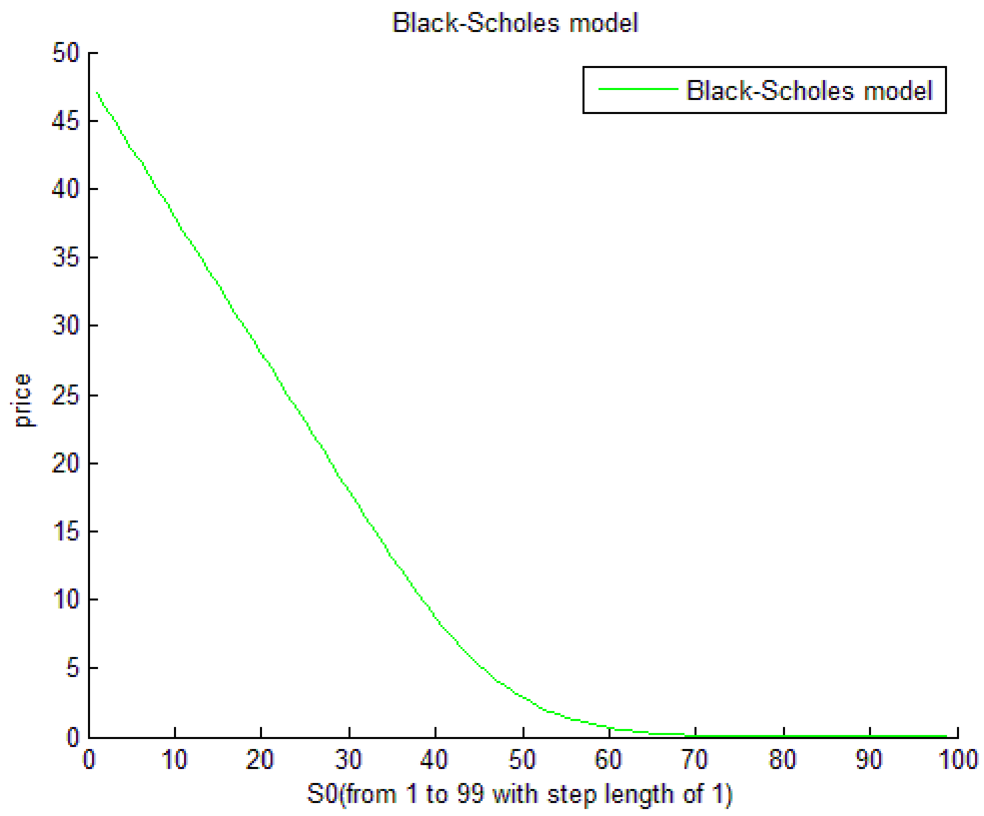

Figure 4

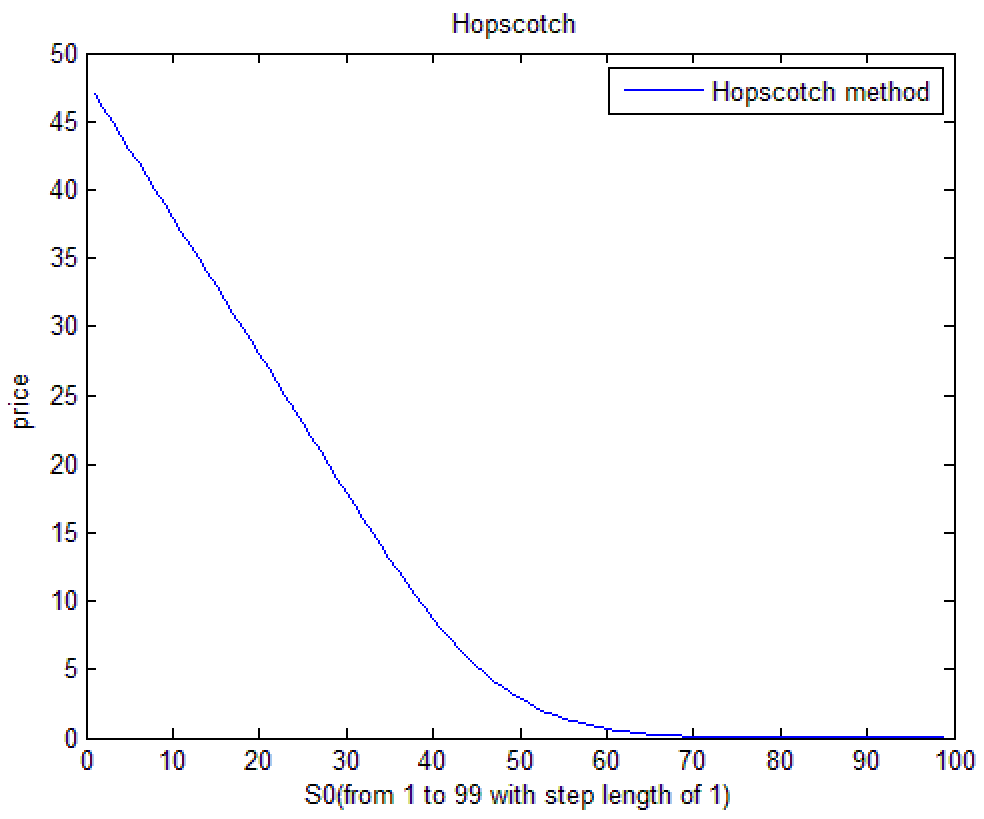

Figure 5

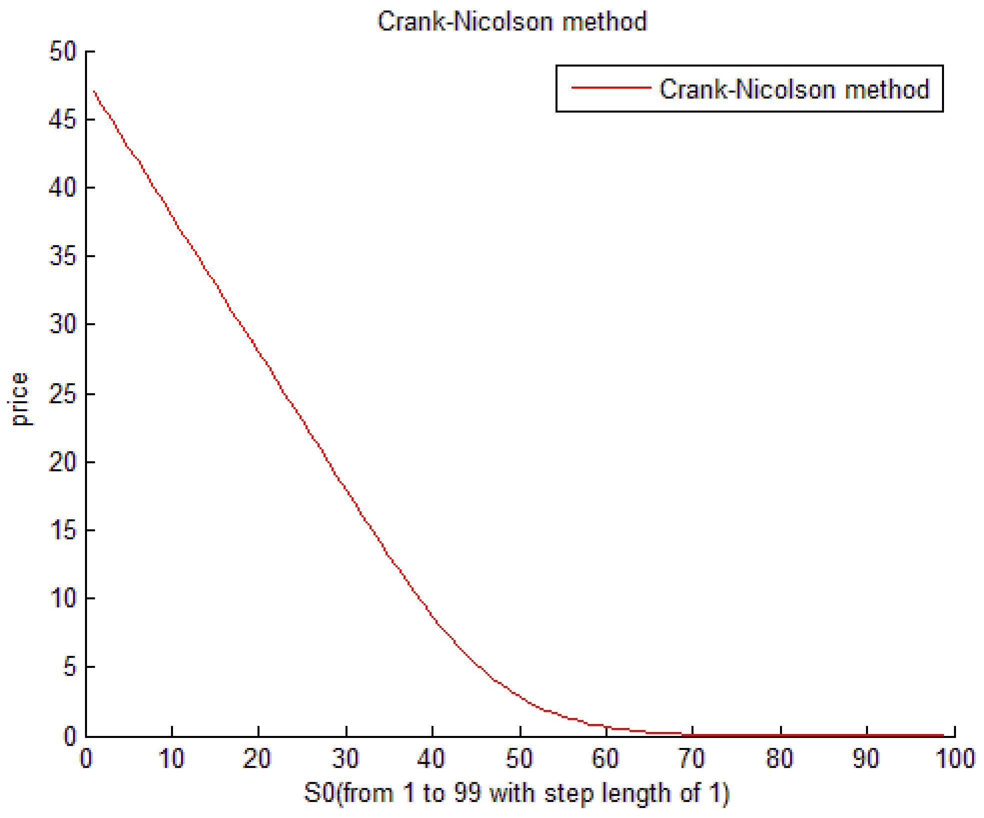

Figure 6

As shown, these three figures look very similar to each other. However, let's turn to a deeper comparing.

4.2.1 Accuracy – absolute error

As we regard Black-Sholes model as a **standard**, we will compare the pricing result of both Hopscotch and Crank-Nicolson methods to the **blsprice**, the more similar the results between Hopscotch and Black-Sholes or Crank-Nicolson and Black-Sholes, the more accuracy it is .

Note: Through our MATLAB applications,we want to give you a clear point of view to see how accurate the two methods are, we choose to find the absolute value of the difference between the pricing results of Black-Sholes model and each of the two methods. (For example: **|Black-Sholes – Hopscotch|** & **|Balck-Sholes – Crank-Nicolson|**). We will call these absolute values as an **'absolute error'**.

Following is the figure that MATLAB generates, it shows the **'absolute errors'** of the two experiments that we want to find out in order to judge and compare the accuracies.

Suppose we set:

$\boldsymbol{S_0}(asset\ price) = 1$ to 99;
$\boldsymbol{K}\ (strike\ price) = 50$;
$\boldsymbol{r}\ \ (risk - free\ interest\ rate); = 0.1$;
$\boldsymbol{\sigma}\ (volatility) = 0.3$;
$\boldsymbol{T}\ (maturity) = 5/12$;
$\boldsymbol{dS}\ (discrete\ step\ of\ asset\ price) = 1$;
$\boldsymbol{dt}\ (discrete\ step\ of\ time) = 5/1200$;
$\boldsymbol{S_{max}}(boundary\ condition) = 100$;

And we will got:

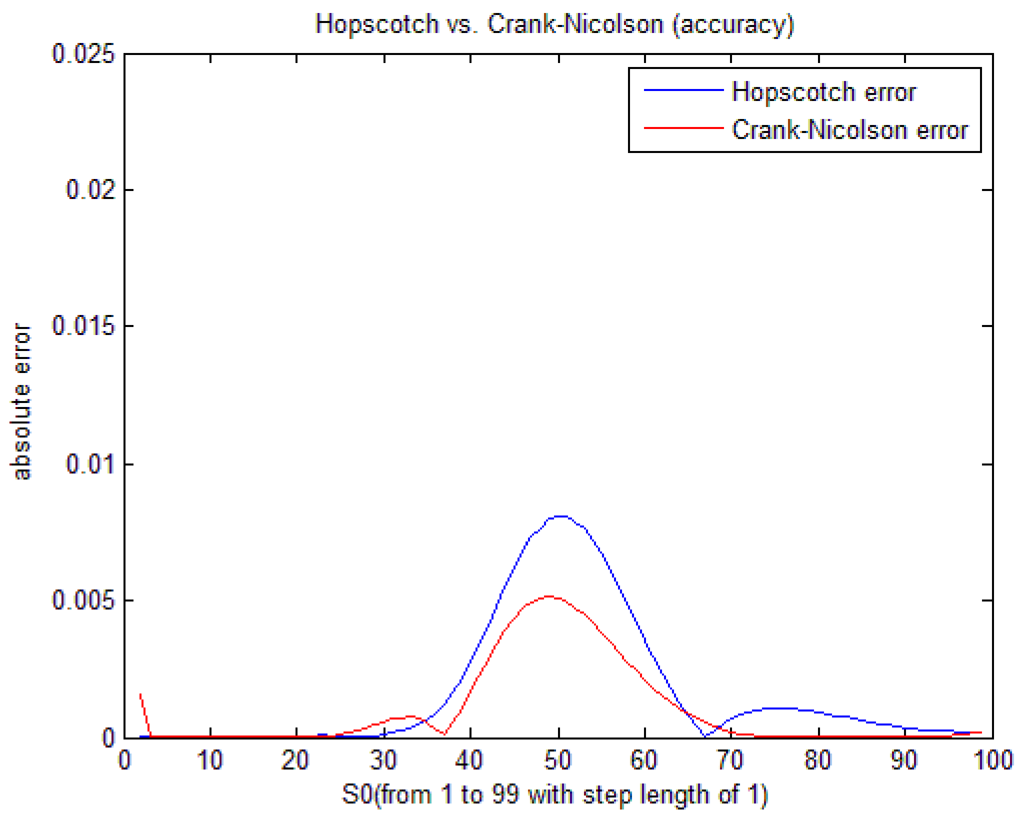

Figure 7

Figure7 shows both methods have quite similar pricing results with the Black-Sholes model (Hopscotch is the blue one, Crank-Nicolson is the red one), which means that both methods have quite small absolute errors, they are apparent smaller than 0.01, the red one (Crank-Nicolson) is even smaller than 0.005. More careful analysis shows that Crank-Nicolson pricing had generate a smaller 'absolute error' than Hopscotch pricing, thus, C-N method is more bit accurate than the Hopscotch method.

4.2.2 Efficiency -- CPU time

After finding out which method is more accuracy, it will be also interested and necessary to know which method is more efficient. That is to find out which method need less CPU time when pricing an option.

We set the same us above:

$\boldsymbol{S_0}(asset\ price) = 1$ to 99;

$\boldsymbol{K}\ (strike\ price) = 50;$

$\boldsymbol{r}\ \ (risk - free\ interest\ rate); = 0.1;$

$\boldsymbol{\sigma}\ (volatility) = 0.3;$

$\boldsymbol{T}\ (maturity) = 5/12;$

$\boldsymbol{dS}\ (discrete\ step\ of\ asset\ price) = 1;$

$\boldsymbol{dt}\ (discrete\ step\ of\ time) = 5/1200;$

$\boldsymbol{S_{max}}(boundary\ condition) = 100;$

And we will got:

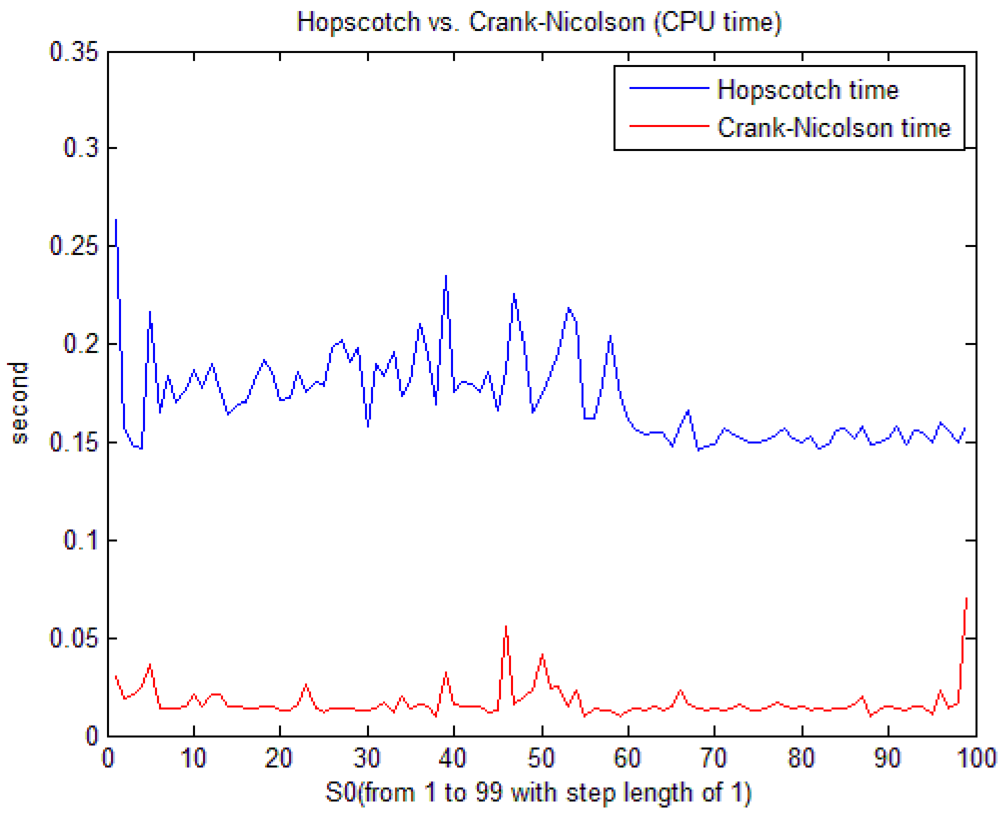

Figure 8

Obviously, we can see that Crank-Nicolson method saves a lot more CPU-time than the Hopscotch method, even though we have to solve a system of equations simultaneously.

5. Conclusion

In this report, we start with an introduction of the numerical approximation of derivatives and apply them to solve the Black-Scholes Partial differential equation. The basic way of solving PDE are use explicit scheme and implicit scheme, there are also two method called Hopscotch method and Crank-Nicolson method which mixed the explicit and implicit scheme to enhance the accuracy of the result they approximate.

Based on this we created the applications of these schemes and methods in MATLAB. By comparing these applications, we found that it is easy to apply the explicit scheme to solve the Black-Scholes PDE. However, we have to suffer the numerical instability of explicit scheme when using it. The Hopscotch method and Crank-Nicolson method integrated the advantage of fully implicit and explicit schemes. The two methods ensure a fairly accurate result for the users, but by comparing the CPU time we found that the Crank-Nicolson method can save more computational time than Hopscotch method even though we have to solve a set of equation simultaneously.

6. Bibliography

[1]Jan R. M. Röman, *Lecture notes in Analytical Finance I*, Mälardalen University, Sweden
August 27, 2010
[2] P. Brandimarte, *Numerical method in finance and Economics A MATLAB based introduction*, 2nd ed., John Wiley & Sons, inc., New Jersey, 2006.
[3] *Financial Toolbox User's Guide*,2nd ed, The MathWorks,inc.,USA,1999.
[4] The MATLAB help file archives, Matlab R2008b, Mathworks.