MATH942: Numerical Methods in Finance, Part 2

Ruben Traicevski 6790021

University of Wollongong

In the following content, common and simple numerical methods used for option pricing are presented, whose overall goal is a proof of concept regarding their power, which is shown in detail, to price vanilla European options in alternative ways, in preparation for pricing more exotic types of derivatives. Specifically, we present the Monte Carlo Method for Option Pricing first due to its very simplistic, yet time consuming, implementation to accurately price options. Moreover, the Monte Carlo Method also serves as an excellent starting point to introduce the GBM SDE, which is one of the foundational pillars of the BSM model, alongside highlighting some concepts typically found in the 'Martingale approach'. Secondly, Finite Difference Methods are analysed through the scope of the Explicit, Implicit and Crank Nicolson marching schemes. This includes their formation, implementation and subsequent consistency and stability analysis, which together are able to ensure convergency. Here, the Discrete Fourier Transform is also introduced and advocated for its use, since it will enable us to derive specific characteristics of the schemes that we otherwise could not have in such a strict sense.

1. Introduction:

The standard Black Scholes Merton Partial Differential Equation is given below,

$$\frac{\partial V}{\partial t}+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{dS}=rV, \qquad 0<t<T \ \ and \ \ 0<S<\infty, \qquad (1)$$

With the following boundary conditions corresponding to vanilla European options whose underlying's pay no dividends,

$$Euro\ Call\begin{cases} Call\colon C_{euro}(S,T)=\max(S-K,0)\ \ for\ all\ S \\ C_{euro}(0,t)=0\ for\ all\ t \\ C_{euro}(S,t)\sim S\ as\ S\to\infty, \qquad for\ fixed\ t \end{cases}, \qquad (1.1)$$

$$Euro\ Put\begin{cases} Put\colon P_{euro}(S,T)=\max(K-S,0)\ \ for\ all\ S \\ C_{euro}(0,t)=Ke^{-r(T-t)}\ for\ all\ t \\ C_{euro}(S,t)\sim 0\ as\ S\to\infty, \qquad for\ fixed\ t \end{cases}, \qquad (1.2)$$

Geometric Brownian Motion is described below by the following Stochastic Differential Equation,

$$dS_t = \mu S_t + \sigma S_t dW_t, \qquad (2)$$

With the following,

$$W_t \sim \mathrm{N}(0, \mathrm{t}), \qquad Wiener\ Process$$

And specifically, μ is used to model deterministic trends, while σ models' unpredictable events during the random path.

The Solution, shown below is well known and verified,

$$ln\frac{S_t}{S_0} = \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t, \qquad (2.1)$$

$$S_t = \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t + ln(S_0)\right)$$

$$E[S_t] = S_0 e^{\mu t}$$

$$Var[S_t] = S_0^2 e^{2\mu t}(e^{\sigma^2 t} - 1)$$

2. Monte Carlo Method for Option Pricing

The method itself is refers to an extremely broad class of computational, and usually time consuming, algorithms that are performing repeated random sampling often hundreds of thousands of times to generate a distribution of simulated results. In our case we are using the theory of risk neutral valuation, hence the price of an option is its discounted expected value. Furthermore, we will be using the famously known SDE, (2), under Geometric Brownian Motion.

2.1 Methodology

1. Generate an acceptable number of random possible of underlying asset price paths through simulating the SDE, (2), by using (2.1)
2. Calculate the payoff, in our case we used a call option, $\max(S - K, 0)$, of each underlying asset price path.
3. Average all the payoffs and discount to present to obtain value of the call option.

2.2 Inputs and Results

S0	K	T	r = μ (2)	volatility	Time Step	BS Price
100	100	1	0.01	0.2	100	8.43

$$Volatility: \sigma^2$$

$$Stock\ price\ at\ t = 0: S_0$$

$$Strike\ Price: K$$

$$Interest\ Rate: r$$

$$Expiration: T$$

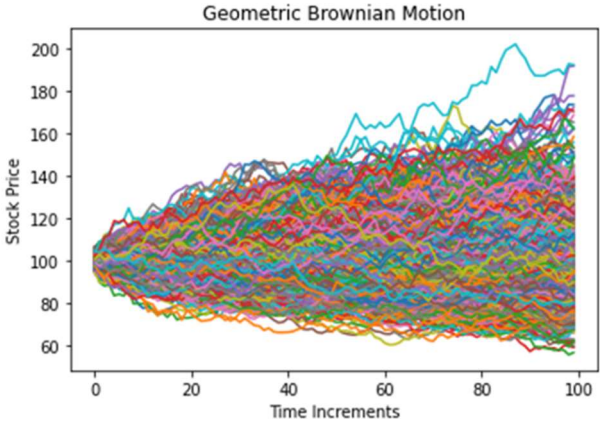


Figure 1. Hundreds of Thousands of randomly generated underlying asset price paths described by GBM.

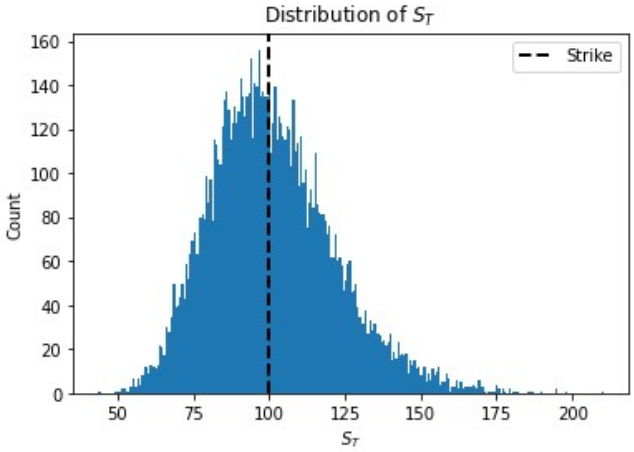


Figure 2. Histogram which describes the distribution of underlying asset price at the 100th time increment. Clearly, we can see it is approximately normally distributed.

At this point, we urge the reader to remember that this is very much a statistical endeavour. Specifically, if we want to achieve a price that is synonymous with the price given by the standard BSM PDE formulas, it is critical that we use enough simulations. Essentially, we want to apply the Law of Large Numbers to ensure that we generate an accurate distribution, in relation to ensuring that the expectation of the distribution formed by all the generated price paths agrees with $E[S_t] = S_0 e^{\mu t}$. Furthermore, generating a distribution is a powerful tool, since the area under the curve of a probability distribution is the probability, meaning we can find the probability that the asset price will finish above the strike price! (in relation to a call option)

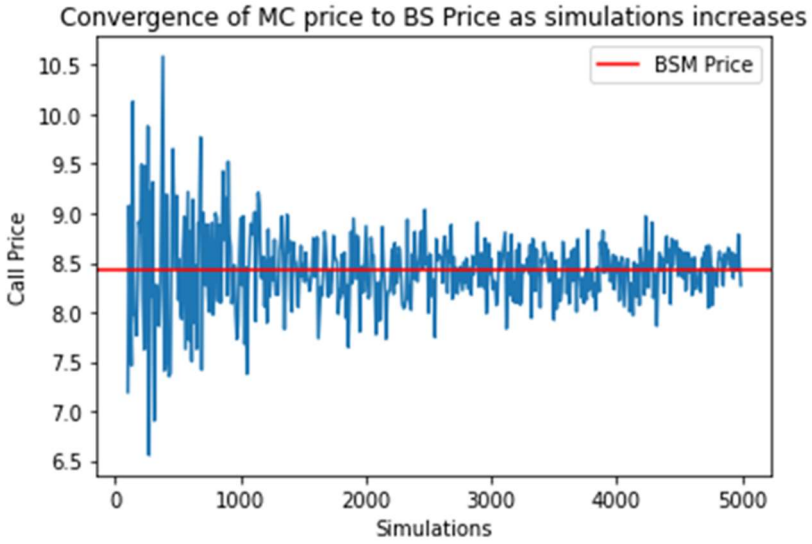


Figure 3. The convergence of the MC price to the BSM Price as we increase the total number of simulations while keeping time increments constant.

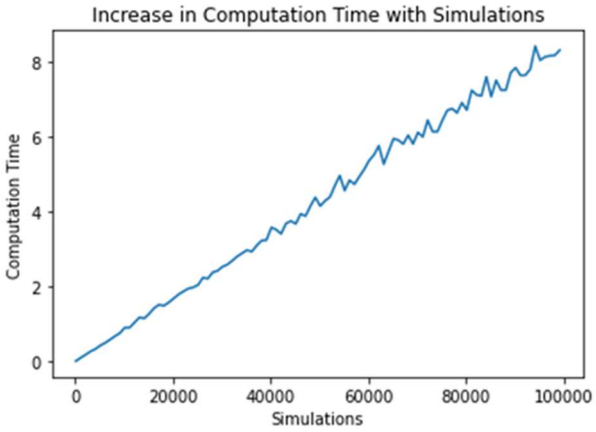
By inspection, it is clear that the price given by the Monte Carlo method, as we increase the number of simulations, is converging upon the 'true' price. After approximately 1,000 simulations the prices settle into a band, which expected in relation to the Law of Large Numbers.

Column1	
Mean	8.419272
Standard Error	0.011082
Median	8.426467
Mode	#N/A
Standard Deviation	0.110819
Sample Variance	0.012281
Kurtosis	9.949271
Skewness	-2.64494
Range	0.740863
Minimum	7.844098
Maximum	8.584961

Compiling all the results into an array and performing some rudimentary summary statistics, we obtained the following results.

Please note, the results of the left are the statistics on the all the prices generated by the Monte Carlo method when we apply for example 1,000 simulations, then 2,000 simulations by using a loop.

This level of accuracy was achieved at 100,000 simulations, and we see that the average of all the prices we generated with the MC Method is almost exactly equal the BSM Price at 2 decimal places. One can even perform a Jarque Bera test to confirm that the kurtosis and skewness matches a normal distribution.



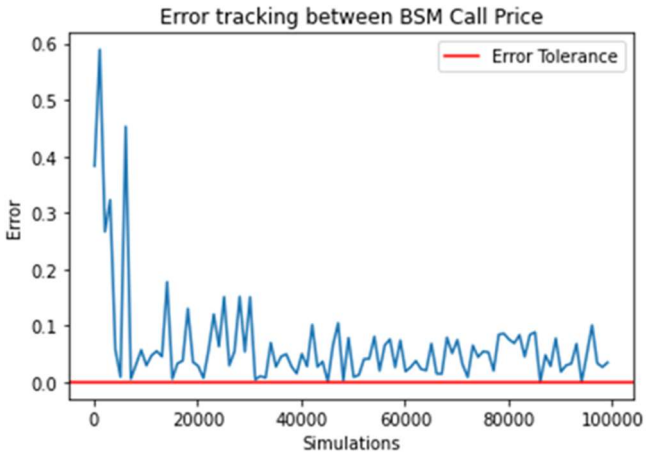


Figure 4. Computational times as function of Simulations when keeping time steps constant, which is exhibiting a Linear trend. Also notice that even with 100,000 simulations we still could not achieve a relative error below 0.004. The Rate of Convergence is extremely poor, with little return in terms of increasing accuracy after approximately 50,000 simulations.

Of course, this level of accuracy and the method itself is not without cost. The 100,000 simulations took a grand total of ~ 428.5299 seconds or ~7.13 minutes. Fitting a straight line through the Computation time, we found that,

$$y = (9E - 05)x - 0.016$$

Thus, with time steps set to 100, we can expect our rate of increase in computational time to be 9E-05 time units, which means every 1 unit increase in simulations, gives a 9E-05 unit computational time increase.

In comparison, a computer using the closed form solutions of the BSM PDE for a Call or Put option can give a price relatively instantly. Clearly, it is much more computationally taxing to use the MC Method over closed form solutions, however, this of course assumes we have solutions. What happens if we need to price much more exotic derivatives? Or even, if we wanted to price vanilla European options whose underlying follows the Heston Stochastic Volatility Model?

3. Finite Difference Methods: Explicit, Implicit and Crank Nicolson

3.1 Discretisation

The following numerical methods presented are specifically, a class of numerical techniques for solving differential equations via approximating derivatives with finite differences. In reference to a PDE, we are approximating the derivatives of the unknown function w.r.t the variables the unknown function is differentiated against. In reference to the BSM PDE, we are discretising the spatial (Price) domain and time domain shown by below,

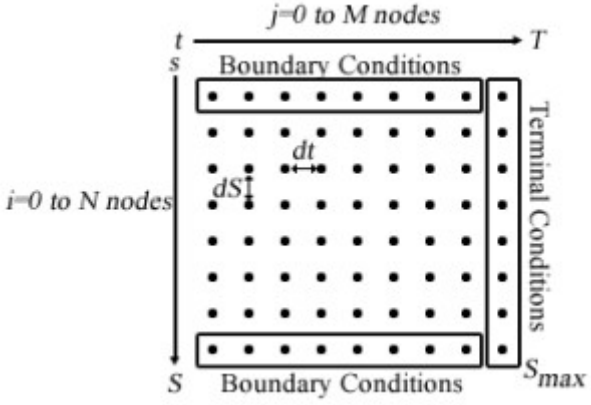
$$S = 0, \Delta S, 2\Delta S, 3\Delta S, \dots, (M-1)\Delta S, S_{max}$$

$$i = 1,2,3, \dots, M-2, M-1$$

$$t = 0, \Delta t, 2\Delta t, 3\Delta t, \dots, (N-1)\Delta t, T_{expiration}$$

$$j = N-1, N-2, N-3, \dots, 2, 1, 0$$

In reference to the boundary conditions 1.1 or 1.2, we essentially form a grid with known Boundary and Terminal conditions, hence J is in reversed range.



3.2 Derivative Approximations

Using (1), we will use the following known difference formulas,

$$Forward: \frac{\partial f}{\partial S} = \frac{f_{i+1,j} - f_{i,j}}{\Delta S}, \qquad \frac{\partial f}{\partial t} = \frac{f_{i,j+1} - f_{i,j}}{\Delta t}$$

$$Backward: \frac{\partial f}{\partial S} = \frac{f_{i,j} - f_{i-1,j}}{\Delta S}, \qquad \frac{\partial f}{\partial t} = \frac{f_{i,j} - f_{i,j-1}}{\Delta t}$$

$$Central: \frac{f_{i+1,j} - f_{i-1,j}}{\partial S}, \qquad \frac{\partial f}{\partial t} = \frac{f_{i,j+1} - f_{i,j-1}}{2\Delta t}$$

$$Second\ derivative: \frac{\partial^2 f}{\partial S^2} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{2\Delta S^2}$$

If one requires the above difference formulas to be more accurate, you will have to trade off efficiency for accuracy.

3.3 Explicit, Implicit and Crank Nicolson Schemes

At this point, we highlight one of the fundamental differences between these schemes. Explicit methods are defined as explicit because they calculate the state of the system at a later time using the state of system at current. Implicit methods require both the state of the system now and state of the system later.

3.3.1 Explicit Scheme

Using use the Backward Difference for the time derivative and Central Difference for the price derivative, alongside the standard second derivative, this ensures that the method is actually Explicit in regard to computing the next future nodal value with value of the node at present. Then in reference to (1) we form,

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

$$f_{i,j} = a_i f_{i-1,j+1} + b_i f_{i,j+1} + c_i f_{i+1,j+1}, \qquad (3)$$

With

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

$$b_i = 1 - \Delta t(\sigma^2 i^2 - ri)$$

$$c_i = \frac{1}{2}\Delta t(\sigma^2 i^2 + ri)$$

3.3.2 Implicit Scheme

Using a Forward Difference in time and Centred Difference in Price we will obtain,

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

$$f_{i,j+1} = a_i f_{i-1,j} + b_i f_{i,j} + c_i f_{i+1,j}, \qquad (4)$$

With the following coefficients,

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

$$b_i = 1 + \Delta t(\sigma^2 i^2 + r)$$

$$c_i = -\frac{1}{2} + \Delta t(\sigma^2 i^2 + ri)$$

3.3.3 Crank Nicolson Scheme

The CN Method uses the combination of both the Implicit and Explicit method, by taking the average of both. This will yield the following,

$$\frac{rf_{i,j-1}}{2} + \frac{rf_{i,j}}{2} = \frac{f_{i,j} - rf_{i,j-1}}{2\Delta t} ri\Delta S\left(\frac{f_{i+1,j-1} - f_{i-1,j-1}}{2\Delta S}\right) + \frac{ri\Delta S}{2}\left(\frac{f_{i+!,j} - rf_{i-1,j}}{2\Delta S}\right)$$
$$+ \frac{\sigma^2 i^2 \Delta S^2}{4}\left(\frac{f_{i+1,j-1} - 2f_{i,j-1} + f_{i-1,j-1}}{\Delta S^2}\right) + \frac{\sigma^2 i^2 \Delta S^2}{4}\left(\frac{f_{i,j} - 2f_{i,j} + f_{i-1,j}}{\Delta S^2}\right)$$

$$-a_i f_{i-1,j-1} + (1 - b_i) f_{i,j-1} - c_i f_{i+1,j-1} = a_i f_{i-1,j} + (1 - b_i) f_{i,j-1} - c_i f_{i+1,j}, \qquad (5)$$

With,

$$a_i = \frac{\Delta t}{4}(\sigma^2 i^2 - ri)$$

$$b_i = \frac{\Delta t}{2}(\sigma^2 i^2 + ri)$$

$$c_i = -\frac{\Delta t}{4} + \Delta t(\sigma^2 i^2 + ri)$$

3.4 LU Decomposition

As highlighted before, since both the Implicit Method and Crank Nicolson Method require both the state of system in the future and at the present, will require matrix methods. However, since we don't know the future values in relation to the standard Ax = B matrix equation, we will employ LU Decomposition since it will only involve the matrix A.

3.4.1 Matrix representation of the Implicit Scheme

Rearranging (4) into the below,

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

Then we can have the following in the form of Ax = B,

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \vdots & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \vdots & 0 & a_{M-2} & b_{M-2} & c_{M-2} \\ 0 & 0 & 0 & 0 & a_{M-1} & b_{M-1} \end{bmatrix} \begin{bmatrix} f_{1,j} \\ f_{2,j} \\ f_{3,j} \\ \vdots \\ f_{M-2,j} \\ f_{M-1,j} \end{bmatrix} = \begin{bmatrix} f_{1,j+1} \\ f_{2,j+1} \\ f_{3,j+1} \\ \vdots \\ f_{M-2,j+1} \\ f_{M-1,j+1} \end{bmatrix} - \begin{bmatrix} a_1 f_{0,j} \\ 0 \\ 0 \\ \vdots \\ 0 \\ c_{M-1} f_{M,j} \end{bmatrix}$$

3.4.2 Matrix representation of the Crank Nicolson Scheme

Rearranging (5) into the below,

$$Af_{j-1} = Bf_j$$

$$A = \begin{bmatrix} 1-b_1 & -c_1 & 0 & 0 & 0 & 0 \\ -a_2 & 1-b_2 & -c_2 & 0 & 0 & 0 \\ 0 & -a_3 & 1-b_3 & -c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & -a_{M-2} & 1-b_{M-2} & -c_{M-2} \\ 0 & 0 & 0 & 0 & -a_{M-1} & 1-b_{M-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 1+b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & 1+b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & 1+b_3 & c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{M-2} & 1+b_{M-2} & c_{M-2} \\ 0 & 0 & 0 & 0 & a_{M-1} & 1+b_{M-1} \end{bmatrix}$$

$$f_i = [f_{1,j}, f_{2,j}, \dots f_{A-1,j}]^T$$

3.4.3 Preliminary Results

Explicit Method

S0	K	r	T	vol	Smax	M	N	Cp	BSM
50	50	10%	5 months	40%	100	100	1000	~6.1078	~ $6.12

Absurd Price with improper grid conditions, on Explicit Method.

S0	K	r	T	volatility	Smax	M	N	Cp
50	50	10%	5 months	40%	100	1000	50	-1.e+38

Implicit Method

S0	K	r	T	volatility	Smax	M	N
50	50	10%	5 months	40%	500	500	1000

Call Price (FD)	Call price BSM	Error	Computation Time
~6.11207	~ 6.1165	0.00443	~16.4617 seconds

Crank Nicolson

S0	K	r	T	volatility	Smax	M	N
50	50	10%	5 months	40%	100	500	1000

Call Price (FD)	Call Price BSM	Error	Computation Time
~6.11276	~ 6.1165	0.00374	~16.2293 seconds

3.5 Lax-Richtmyer Equivalence theorem

A consistent finite difference method for on a well-posed linear initial value problem, the method is convergent if and only if it is stable,

$$consistency + stability \leftrightarrow Convergence$$

3.5.1 Consistency

A finite difference scheme is pointwise consistent with its PDE, the PDE it is defined upon, if the following relationship holds,

$$(Original\ PDE)_i^j - (FD\ Scheme)_i^j \to 0, \qquad as\ \Delta S, \Delta\tau \to 0\ and\ (x_i, \tau_{j+1}) \to (x, \tau)$$

Omitted due to lengthy derivation, for the intents of the report, we can ensure consistency with appropriate number of M and N nodes chosen.

3.5.2 Transforming BSM PDE into the Heat Equation

The following stability analysis is much more easily explained when we convert the BSM PDE into the well-known Heat Equation.

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}, \quad 0 < \tau \leq \frac{2}{\sigma^2} T \ \ and \ \ -\infty < x < \infty$$

3.5.2.1 Explicit, Implicit and Crank Nicolson Schemes on the Heat Equation

Explicit Method

$$U_i^{(n+1)} = rU_{i-1}^{(n)} + (1-2r)U_i^{(n)} + rU_{i+1}^{(n)}$$

Implicit Method,

$$-rU_{i-1}^{(n+1)} + (1+2r)U_i^{(n+1)} - rU_{i+1}^{(n+1)} = U_i^{(n)}$$

Crank Nicolson Method,

$$-rU_{i-1}^{n+1} + 2(1+r)U_i^{n+1} - rU_{i+1}^{n+1} = f_i^n$$

$$\text{Where for all, } r = \frac{\Delta t}{\Delta x^2}$$

3.5.3 Stability Analysis

To sufficiently analyse the stability of the schemes, we will employ the Discrete Fourier Transform to derive conditions for stability in a strict sense.

3.5.3.1 Discrete Fourier Transform

From Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach by Daniel J. Duffy, a method for Stability analysis is to employ the Discrete Fourier Transform on the Finite Difference Scheme's

Let $\mathbf{u} = {}^t(\ldots, u_{-1}, u_0, u_1, \ldots)$ be an infinite sequence of values. Then the DFT (given in Thomas, 1998) is defined as

$$\hat{u}(\xi) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} \mathrm{e}^{-in\xi} u_n \tag{8.30}$$

One can apply the DFT to the FD schemes given above and derive certain results. An example of such derivation is given below,

$$\textbf{\textit{Implicit DFT}}: \frac{1}{\sqrt{2\pi}} \sum_{i=-\infty}^{\infty} e^{di\xi}\left[U_i^{(n+1)}\right] = \hat{U}_i^{(n)}(\xi)$$

$$= \frac{1}{\sqrt{2\pi}}\left(\sum_{i=-\infty}^{\infty} e^{di\xi}\left[-rU_{i-1}^{(n+1)} + (1+2r)U_i^{(n+1)} - rU_{i+1}^{(n+1)}\right]\right)$$

$$= \frac{-r}{\sqrt{2\pi}} \sum_{i=-\infty}^{\infty} e^{di\xi}\left[U_{i-1}^{(n+1)}\right] + \frac{(1+2r)}{\sqrt{2\pi}} \sum_{i=-\infty}^{\infty} e^{di\xi}\left[U_i^{(n+1)}\right] + \frac{-r}{\sqrt{2\pi}} \sum_{i=-\infty}^{\infty} e^{di\xi}\left[U_{i+1}^{(n+1)}\right] \quad (1)$$

Since the sum does not include the index n, we can use the following change of variables to simplify,

$$\frac{1}{\sqrt{2\pi}} \sum_{i=-\infty}^{\infty} e^{-di\xi}\left[U_{i\pm1}^{(n+1)}\right] = \frac{e^{\pm di\xi}}{\sqrt{2\pi}} \sum_{m=-\infty}^{\infty} e^{-dm\xi}\left[U_m^{(n+1)}\right] = e^{\pm d\xi}\hat{U}_i^{(n+1)}(\xi), \qquad m = i \pm 1$$

Applying this to (1),

$$\hat{U}_i^{(n)}(\xi) = -re^{-d\xi}\hat{U}_i^{(n+1)}(\xi) + (1+2r)\hat{U}_i^{(n+1)}(\xi) - re^{d\xi}\hat{U}_i^{(n+1)}(\xi)$$

$$= \hat{U}_i^{(n+1)}(\xi)\left(-re^{-d\xi} + (1+2r) - re^{d\xi}\right)$$

$$= \hat{U}_i^{(n+1)}(\xi)\left(-2rcos(\xi) + (1+2r)\right)$$

$$\frac{\hat{U}_i^{(n+1)}(\xi)}{\hat{U}_i^{(n)}(\xi)} = p(\xi) = \frac{1}{1 + 4r^2 sin^2\left(\frac{\xi}{2}\right)}$$

Therefore, we have that, for errors to remain bounded, we apply,

$$|p(\xi)| = \left| \frac{1}{1 + 4r^2 sin^2\left(\frac{\xi}{2}\right)} \right| \leq 1$$

CN DFT:

Following similar arithmetic involving the DFT will yield a similar result,

$$p(\xi) = \frac{1 - 2rsin^2\left(\frac{\xi}{2}\right)}{1 + 2rsin^2\left(\frac{\xi}{2}\right)}$$

$$|p(\xi)| = \left| \frac{1 - 2rsin^2\left(\frac{\xi}{2}\right)}{1 + 2rsin^2\left(\frac{\xi}{2}\right)} \right| \leq 1$$

$$\rightarrow |p(\xi)| \leq 1 \; for \; any \; r$$

Explicit DFT:

$$p(\xi) = 1 - 4r^2 sin^2\left(\frac{\xi}{2}\right)$$

$$|p(\xi)| = \left| 1 - 4r^2 sin^2\left(\frac{\xi}{2}\right) \right|$$

$$\left| 1 - 4r^2 sin^2\left(\frac{\xi}{2}\right) \right| \leq 1$$

This condition is only satisfied when $\boldsymbol{r = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}}$

This implies that the Explicit scheme is conditionally stable, meaning r must be $\leq \frac{1}{2}$ to ensure errors remain bounded, while the Implicit and CN Schemes are unconditionally stable.

3.6 Benefits of Unconditional Stability

Apart from unconditional stability essentially ensuring convergence, assuming consistency, which is easily able to be achieved, the main benefit of unconditional stability comes from able to reduce the number of nodes in the grid. This is of particular importance because, we note that matrix methods such as LU Decomposition are computationally taxing, whereas the Explicit method is easily implemented and easily handled by the computer.

3.6.1 Robustness of Implicit and Crank Nicolson Schemes

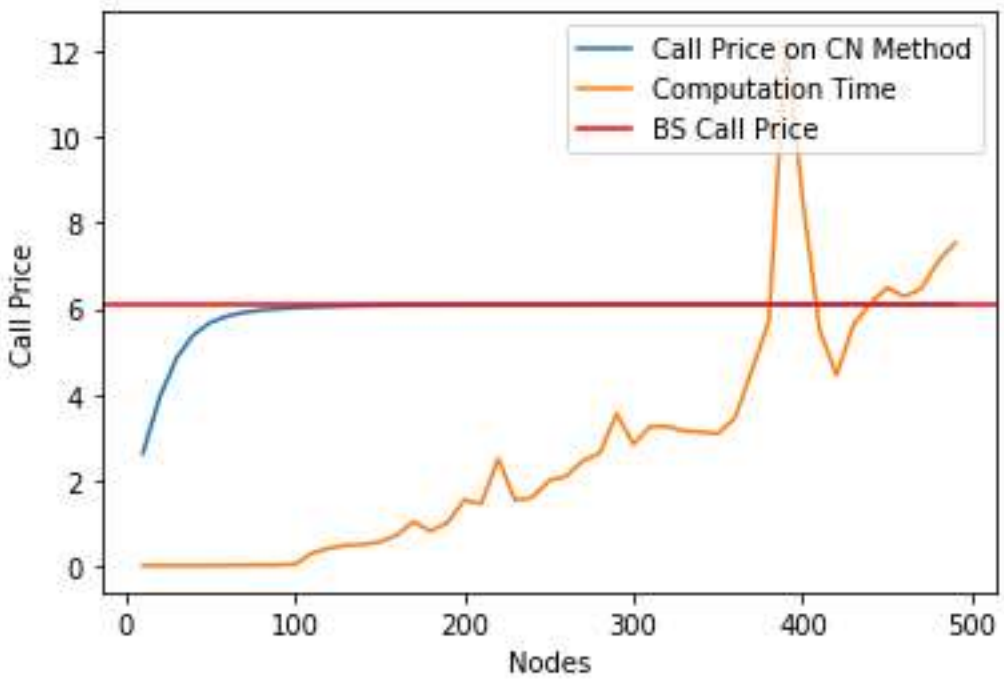
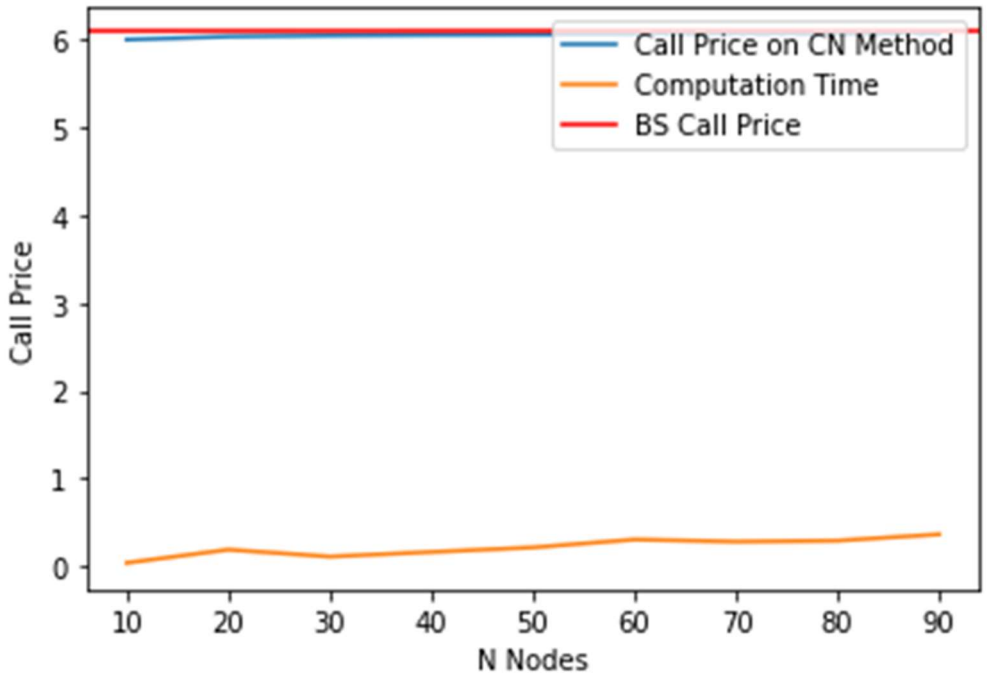


Figure 5: Recording computation time and Price approximations by the Crank Nicolson Schemes by varying N and M nodes in tandem

It is clearly evident that we achieve a satisfactory level of relative error between 100 to 200 nodes in both N and M. It is also clear that the returns, in terms of better levels of accuracy, greatly diminish as we increase the amount nodes, at a cost of computation time. This highlights that we are essentially wasting an advantage of 'being' unconditionally stable by being negligent with the number of nodes.



The figure above is what we claim is an efficient use of computational resources. Holding the M nodes at 150, we can clearly see an acceptable level of relative error, and a much lower computational time, that does not break above 1 second.

3.7 Advantages and Disadvantages

	Advantages	Disadvantages
Implicit	Unconditionally stable, allowing use of larger increment time steps. Fewer time steps needed to carry out calculation over an interval	Slower Computation time in versus Explicit Method Needs simultaneous solution in each time step. Computationally expensive
Crank Nicolson	Unconditionally stable, allowing use of larger increment time steps. Fewer time steps needed to carry out calculation over an interval	Slower Computation time in versus Explicit Method Needs simultaneous solution in each time step. Computationally expensive
Explicit	Easily implemented. Requires only the present state of the system to calculate future state of system. Computationally inexpensive relative to Implicit and Crank Nicolson Schemes	Unconditionally stable. Much more sensitive to changes in number of M and N nodes. If one wishes to use the Explicit Scheme, we implore them to use the condition that at all times $\boldsymbol{r = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}}$ to ensure we in a region of stability

4. Final Remarks

The methods presented above of course may seem redundant when highlighting that closed form solutions already exist for vanilla European Options. However, we stress that their importance cannot be understated when attempting price much more complicated and exotic derivatives. The Monte Carlo method is clearly applicable when trying to price derivatives who's underlying may follow stochastic processes more difficult to rigorously define as opposed to GBM. Likewise, we may develop a PDE to accurately describe the price of a derivative, but cannot obtain a closed form solution, in which case, Finite Difference Methods become a prime candidate for an applicable numerical method.