Comparing Accuracy and Efficiency of Numerical Methods for Solving the Black-Scholes Equation in European and American Options

Yamini Ananth, Vincent Zhao

April 2022

Abstract

This project presents analytical and numerical problem solving methods for the Black-Scholes equation, which is used for option pricing. We discuss the analytic solution and the discretization of the PDE, along with two types of boundary conditions. We compare finite difference, explicit, implicit, and stochastic methods for both European and American options. We furthermore discuss when stochastic solutions may be more appropriate.

1 Introduction

The Black-Scholes equation models the future price of a stock or option. Options give their buyer the right to buy or sell an asset at a specific 'exercise price' (E) on or before a specified date, the 'expiry time' (T). The asset has its own value (S), which changes based on a certain volatility (σ).

European options can only be executed on their expiry time, whereas American options can be exercised at any time before the expiry time. Options themselves can also be bought and sold based on their price, which fluctuates based on the time and the asset's current value ($V(S,t)$). Call options grant the owner the right to buy shares from the buyer through the expiry date, whereas put options allow the owner to sell. For this paper, we will consider only call options, though by put-call parity, we can extend our findings to the put option for the European model.

The Black-Scholes Equation, shown below, can be used to calculate the price of an option.

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

The analytic solution to Black-Scholes for the European Option is as follows

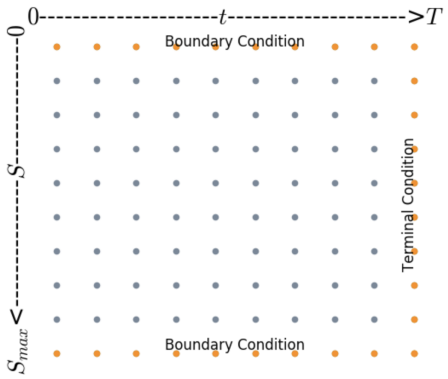


Figure 1: Graphic generated with code from [6]

(derivation in Appendix A). We will use this true solution to evaluate our numerical methods in Section 3.

$$\begin{aligned} V(\tau) =& e^{\ln S(\tau)}\Phi\left(\frac{-\ln K + \ln S(\tau) + \left(r - \frac{1}{2}\sigma^2\right)\tau + \sigma^2\tau}{\sigma\sqrt{\tau}}\right) \\ &- Ke^{-r\tau}\Phi\left(\frac{-\ln K + \ln S(\tau) + \left(r - \frac{1}{2}\sigma^2\right)\tau}{\sigma\sqrt{\tau}}\right) \end{aligned} \tag{2}$$

2 Discretizing the Black Scholes Equation

2.1 Grid

We must discretize in space and time, so we will divide time into N subintervals and space into M subintervals, resulting in an $N \times M$ grid that represents the space $[0, S_{max}] \times [0, T]$

$$0 < t < T, t = 0, \Delta t, 2\Delta t, \ldots, N\Delta t$$
$$0 < s < S_{max} = 0, \Delta s, 2\Delta s, \ldots, M\Delta s$$

In the original solution of the Black-Scholes equation, the solution f^* is defined for the continuous space; however, the approximate solutions are only defined on the nodes. We will define this approximation as

$$f_{i,j} = f(i\Delta s, j\Delta t)$$

In our implementation, we set parameter S_{max} to be four times the exercise price, following the simplest precedent. By adjusting parameters M, N and S_max, we can influence the error of the discretization such that $f_{i,j} \to f^*$.

2.2 Boundary Conditions

Next, we can consider the moment of expiry for a call option. There are two possibilities:

1. $S > E$, then we should exercise the option and profit $S - E$.
2. $S < E$, we should not exercise the option, we would lose money, so we are better off letting it expire worthlessly.

As a result, we can write the value of a call option at expiry as

$$C(S, T) = max(S - E, 0)$$

We can take this as our initial condition, such that

$$C(S, 0) = max(S - E, 0)$$

which yields Dirichlet boundary conditions

$$\begin{cases} C(0, t) = 0 \\ C(S, t) = S & S \to \infty \\ C(S_{max}, t) = S_{max} - Ke^{-r(T-t)} \end{cases}$$

We will use these Dirichlet boundary conditions for the European models we explore. There are advantages to using Neumann boundary conditions, namely that they are the same for both call and put options, and they are also more stable around the boundaries. However, since we are only looking at call options. Furthermore, it has been found that global error with the linearity condition applied and the Dirichlet boundary condition has minimal global error [5].

3 European Option - Numerical Methods for Solving the BS BVP

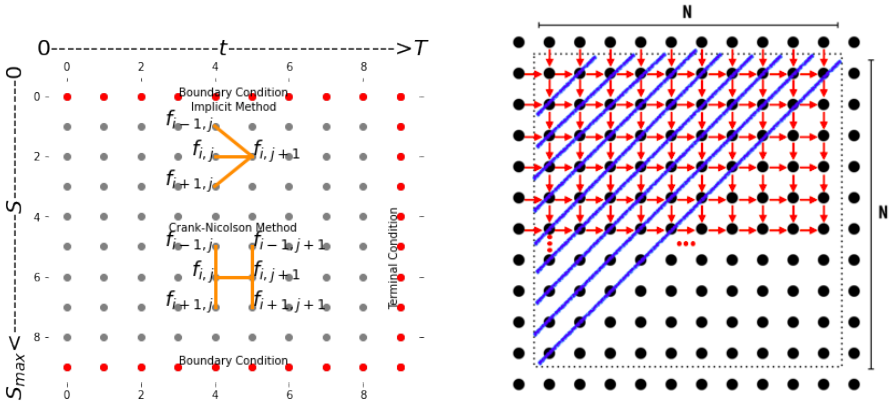
We will consider three methods, Backwards Euler, Successive Over-relaxation, and Crank Nicholson, and the Monte Carlo method.

3.1 Backwards Euler Method (explicit)

3.1.1 Derivation

We derive this method using a backward difference approximation for t and a central difference for t, expanding at $(i\Delta s, j\Delta t)$ (Figure 2).

$$\frac{f_{i,j} - f_{i,j-1}}{\Delta t} + ri\Delta s\frac{f_{i+1} - f_{i-1,j}}{2\Delta s} + \frac{(i\sigma\Delta s)^2}{2}\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta^2 s}$$



(a) Explicit,C-N (b) SOR

Figure 2: Grid traversals for numerical methods. Figures generated with minor adjustment to [6]

We can rearrange to get

$$\alpha_i f_{i-1,j} + (1-\beta_i) f_{i,j} + \gamma_i f_{i+1,j} = f_{i,j-1} \tag{3}$$

$$\begin{aligned}
\alpha_i &= \frac{i\Delta t}{2}(i\sigma^2 - r) \\
\beta_i &= -\Delta t(i^2\sigma^2 + r) \\
\gamma_i &= \frac{i\Delta t}{2}(i\sigma^2 + r)
\end{aligned}$$

These recursive formulas can be translated to sparse matrices, which we solve using the recursive formulas directly (as opposed to matrix operations) for improved efficiency.

3.1.2 Stability

The backward numerical method can be written as:

$$U_{n+1} = U_n + \Delta t f(t_{n+1}, U_{n+1}).$$

We can obtain $R(z)$ by rewriting U_{n+1} as:

$$\begin{aligned}
U_{n+1} &= \frac{1}{1-\Delta t\lambda} U_n \\
&= \frac{1}{1-z} U_n
\end{aligned}$$

$$R(z) = \frac{1}{1-z}$$

and absolute stability requires

$$|R(z)| \leq 1 \leftrightarrow |1-z| \geq 1$$

Which means backward Euler method is stable for any choice of $Delta_t$ for $\lambda < 0$.

3.1.3 Convergence

The local truncation error of the Euler method is bounded by $O\left(h^{p+1}\right)$ and the global error is bounded by $O\left(h^{p}\right)$. This method converges for any $\Delta t, \lambda$.

3.2 Successive Over-Relaxation Method

3.2.1 Derivation

In this iterative method, we make use of updated values of x as soon as they become available. We derived this method in class and we can use a similar scheme here. To implement this method, we can follow a similar matrix-based procedure as demonstrated in class as well. [3]

3.2.2 Stability

Successive Over-relaxation (SOR) can be derived from the Gauss-Seidel iteration method. It is noticed that Gauss-Seidel is stable only when the system is diagonally dominant, which means the absolute value of each diagonal element is at least as large as the sum of the absolute values of the remaining entries in the same row. Then, the stability of SOR can be proved by the stability of Gauss-Seidel. We showed in class that SOR is stable for $w > 1$ [3].

3.2.3 Convergence

This method is only considered convergent where the relaxation parameter satisfies $0 < \omega < 2$, and it converges faster than the Gauss-Seidel method only where $1 < \omega < 2$; where ω=1. To choose ω optimally, we can use

$$\omega = \frac{2}{1 + \sqrt{1 - \rho_j^2}}$$

where ρ_j which is the Jacobi iteration's spectral radius (i.e. if Gauss-Seidel and SOR for the optimal ω produce similar results, then $\omega_{optimal} \approx 1$). [2]

3.3 Crank-Nicholson Method

3.3.1 Derivation

The Crank-Nicholson method, which uses elements of both an implicit and explicit scheme, can be applied to the Black-Scholes equation to achieve a higher accuracy and stability than a fully implicit scheme.

The Crank-Nicholson method prices three left-side nodes based on the values of the right-side nodes by solving a series of M-1 equations. We use a central difference on t and S, discretizing the B-S equation at $(i\Delta s, (j+\frac{1}{2})\Delta t$ and using the average of the first derivative on $(i\Delta s, (j+1)\Delta t$ and $(i\Delta s, (j)\Delta t$ to approximate the first derivative. Similarly, we can also approximate the second derivative.

$$\begin{aligned}
\frac{\partial f}{\partial t}(S_i, t_{j+\frac{1}{2}}) &= \frac{f_{i,j+1} - f_{i,j}}{\Delta t} \\
\frac{\partial f}{\partial t}(S_i, t_{j+\frac{1}{2}}) &= \frac{1}{2}(\frac{\partial f}{\partial t}(S_i, t_{j+1}) + \frac{\partial f}{\partial t}(S_i, t_j)) \\
\frac{\partial^2 f}{\partial t^2}(S_i, t_{j+\frac{1}{2}}) &= \frac{1}{2}(\frac{\partial^2 f}{\partial t^2}(S_i, t_{j+1}) + \frac{\partial^2 f}{\partial t^2}(S_i, t_j))
\end{aligned}$$

We can discretize, plug these derivatives into the BS equation to get the following equations:

$$-\alpha_i f_{i-1,j} + (1-\beta_i) f_{i,j} - \gamma_i f_{i+1,j} = \alpha_i f_{i-1,j+1} + (1+\beta_i) f_{i,j+1} + \gamma_i f_{i+1,j+1} \quad (4)$$

$$\begin{aligned}
\alpha_i &= \frac{i\Delta t}{4}(i\sigma^2 - r) \\
\beta_i &= -\Delta t(i^2\sigma^2 + r) \\
\gamma_i &= \frac{i\Delta t}{4}(i\sigma^2 + r)
\end{aligned}$$

These recursive formulas can be translated to sparse matrices, which we solved directly using numpy's spsolve method.

3.3.2 Stability

Crank Nicholson is stable when

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

We can reason through this since, when this condition holds true, the values of F_i get smaller and smaller, so the algorithm converges. It has been shown that, for all values of $p, \sigma, \Delta t, \Delta s$, Crank-Nicholson is stable (unlike the implicit methods) [2].

3.3.3 Convergence

The rate of convergence for the Crank-Nicholson method is directly related to the truncation error that is associated with the partial derivative approximations. Thus, it should converge at a rate of $\mathcal{O}(\Delta t^2 + \Delta s^2)$, faster than the explicit and implicit methods. [3]

3.4 Monte Carlo Method

We used the Feyman-Kac theorem and defined an equivalence between the stochastic and PDE representations to generate an analytic, closed form solution to the Black-Scholes equation (Appendix A). However, we can also simulate solutions to the stochastic differential equation itself using Monte Carlo methods. Here, we will solve the following:

$$dS = \mu S dt + \sigma S_t d\omega \tag{5}$$

where $\omega = \omega(t)$ is the standard normal random distribution and $\mu and \sigma$ are average growth rate and volatility (assumed constant).

In the case of the European option, we carry out the simulation until maturity, then apply the terminal payoff function and calculate their mean values. We will discount mean values to their present value to finally retrieve the option's value.

We implemented this using a Tensorflow decorator to make the runtime faster.

4 American Option - Numerical Methods for Solving the BS BVP

As mentioned previously, American options can be exercised at any time leading up to their expiry, unlike the European option. To apply BS to American options and solve with numerical methods, certain accommodations must be made.

4.1 Assumptions and Boundary Conditions

Because of the early exercise option, we have a free boundary on our discretized grid. For any time t, there is a key price, which we can call $S_f(t)$. For this key price, there are two possibilities:

$$\begin{cases} S(t) \leq S_f(t) & f(S,t) = K - S \\ S(t) > S_f(t0 & f(S,t) > (K-S)^+ \end{cases}$$

The set of points on which the exercise of the American option is the early exercise (free) boundary, and this is a curve $S = S^*(t)$ for a simple option. We can denote this boundary $B(t)$.

By modeling this situation using a Black-Scholes equation for a an asset V, yield D, we can get the equation

$$V_t + \frac{1}{2}\sigma^2 S^2 V_{ss} + (r - D)SV_s - rV = 0$$

This has the following boundary conditions:

$$\begin{cases} V(S,t) = max(S - K, 0) \\ B(T) = max(K, \frac{rK}{D} \\ V(0,t) = 0 V(B(t),t) = B(t) - K \\ V_s(B(t),t) = 1 \end{cases}$$

which we can use when applying a similar procedure as in the European option to derive our numerical methods.

4.2 SOR

For our implementation of the SOR method for the American option, we followed a similar strategy to our method for the European option. However, when we implemented the European option, we used fixed numbers of iterations to improve computation speed, and also as a parameter to control. Here, we allowed the method to iterate until a tolerance threshold was satisfied.

Using this SOR method, we were able to achieve the following results:

S	σ	T	BS Closed	AM$_S OR$	Early-Exercise Value
2	0.2	1	0.000003	0.000003	-0.000003
2	0.2	2	0.001454	0.001196	-0.001118
2	0.4	1	0.009010	0.008405	-0.001028
2	0.4	2	0.077005	0.070752	-.0017150
5	0.2	1	0.663484	0.627691	-0.004915
5	0.2	2	1.085968	1.007536	0.004552
5	0.4	1	1.015923	0.982618	-0.015207
5	0.4	2	1.526494	1.453893	-0.01229
8	0.2	1	3.476583	3.397119	0.010627
8	0.2	2	3.911142	3.753795	0.012077
8	0.4	1	3.558124	3.482719	0.017097
8	0.4	2	4.088889	3.940458	0.033799

As you can see, the American SOR method here is actually quite similar to the European option in many cases, although it does offer an early-exercise value in many cases, particularly when our starting asset S=8 was greater than our strike price of 5.

We did not expect to see situations where our American SOR method yielded a result which was worse than the closed form European call solution, since we

had tried to implement our American SOR method in a way such that it would either match or exceed the European option. We would like to spend more time investigating this issue when we have the time.

4.3 Longstaff-Schwartz Least Squares Monte Carlo method

As we discussed when deriving the analytic solution (Appendix A), the price of a stock (S) is assumed to follow geometrical Brownian motion. We can use this to implement the Longstaff-Schwartz Least-Squares Method (LSLS), which takes into account the early exercise option.

In this method, N random paths are generated $(S_n^k, t_n), 1 \leq k \leq N, t_n = ndt$. The future expectation is replaced with a least squares regression, and valuation is performed by rolling back on these paths.

The price of the option is calculated as the average value of all the discounted payoffs, which is just the average of the row sums in our matrix form.

4.3.1 Convergence

The first three Laguerre polynomials (ie degree=3) is suffient to obtain effective convergence of the Monte Carlo Least Squares algorithm in this scenario of a vanilla American call option. Convergence is shown more extensively by Stentoff [4].

4.3.2 Implementation and Analysis

We used an object-oriented implementation of the Longstaff-Schwartz method, using code from [1].

We generated the following results for a series of parameter schemes, and compare them to the closed form European call option as a point of reference: Early Exercise Value refers to the difference between the MC-LS price and the European Call method. We hold K=5, r=0.06 constant.

S	σ	T	BS Closed	MC-LS	Early-Exercise Value
2	0.2	1	0.000003	0.000000	-0.000003
2	0.2	2	0.001454	0.000336	0.001117
2	0.4	1	0.009010	0.007982	-0.001028
2	0.4	2	0.077005	0.075290	-0.001715
5	0.2	1	0.663484	0.668399	0.004915
5	0.2	2	1.085968	1.090520	0.004552
5	0.4	1	1.015923	1.000716	0.015208
5	0.4	2	1.526494	1.514198	0.012295
8	0.2	1	3.476583	3.487210	0.010627
8	0.2	2	3.911142	3.923219	0.012077
8	0.4	1	3.558124	3.575221	0.017097
8	0.4	2	4.088889	4.122688	0.033798

4.4 Comparision of Methods

From the two tables generated previously, we can calculate the relative error of our Monte Carlo LS method and our American SOR implementation, as in Figure 4.4, and also consider where one method might outperform the other. To calculate the difference of early exercise, we find the difference between the early exercise of SOR-Monte Carlo.

S	σ	T	BS Closed	Rel. Err.	Diff. of Early Exercise
2	0.2	1	0.000003	–	-0.000003
2	0.2	2	0.001454	0.7190	-0.0111880
2	0.4	1	0.009010	0.05032	-0.001028
2	0.4	2	0.077005	0.06413	-0.001715
5	0.2	1	0.663484	0.06485	0.004915
5	0.2	2	1.085968	0.08236	0.004552
5	0.4	1	1.015923	0.01841	0.015208
5	0.4	2	1.526494	0.04147	0.012295
8	0.2	1	3.476583	0.02651	0.010627
8	0.2	2	3.911142	0.045134	0.012077
8	0.4	1	3.558124	0.02656	0.017097
8	0.4	2	4.088889	0.046245	0.033798

We observe that in all parameter schemes, the two methods yielded a similar response. However, somehow, in the situation where S is less than K, the Monte Carlo method yields a better early exercise. This is with a very small amount of sample data, but it offers an interesting situation wherein the least squares method might be preferred. Additionally, the Monte Carlo methods took approximately 9.23 seconds to calculate values for all 12 parameter schemes, whereas the SOR method took about 78 seconds. Considering the closeness of the two results, the speed of the least squares MC method is an advantage.

5 Discussion

5.1 European Option

The price of the European call option was solved based on the Black-Scholes equation. We compared four different methods and calculated the convergences of our finite difference methods using different grid numbers of time (see Figures).

At first glance, all methods yield graphs that look fairly similar to the closed form analytic solution to BS, which you can see in figure **??**.

As we examined this further, we noticed that our Monte Carlo returned results with the most error. Although the difference is hard to capture in figure **??**, backward Euler returned the worst among C-N, SOR and Euler methods, which

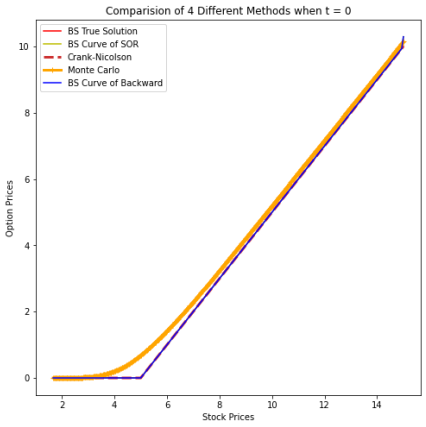


Figure 3: Comparison of 4 Different Methods when t = 0

is expected since it does not take iteration into consideration, it just loops from the last value to the first value.

In all iterative methods, the total number of iterations has a great impact on the convergence of the plot, which is also an important part we want to cover in our project. We chose to study the SOR method in particular. We first set the number of iteration as 300, but found it to be too time-consuming to study feasibly. The main loop of the algorithm will take N_space $\times$ N_time times in total, also, since we have three schemes and 4 N_times for each scheme, the whole loop will take almost an hour to finish. Then we changed our number of iteration to 100, which will return the desired values within 10 minutes.

We further analysed the SOR method with three parameter schemes with different rates of return and volatility are shown in 5.1:

Scheme	r	*sigma*
1	0.3	0.06
2	0.06	0.03
3	0.6	0.3

Each of these schemes was analyzed with four different sized grids, where SOR_time $= [200, 500, 900, 1500]$.

After plotting the graphs(shown as below), it is noticed that the results of scheme 1 convergent to the 1st order convergence when the number of iteration is not a controlling parameter. When risk free rate of return and volatility change, it will take more steps than the number we set to reach the 1st order convergence.

For scheme 2 and scheme 3, four results $||True - Sol||$ calculated by different grid numbers are fairly close to each other. The result of scheme 2 ranges from

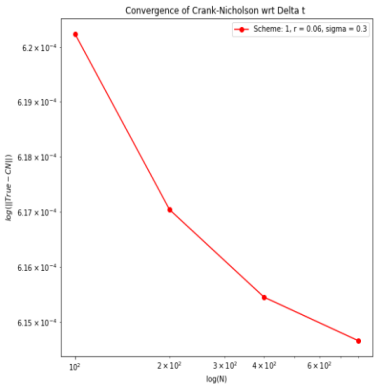


Figure 4: C-N Scheme 1

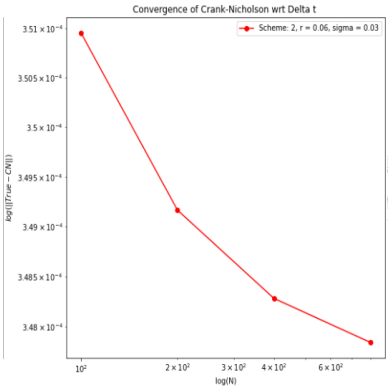


Figure 5: C-N Scheme 2

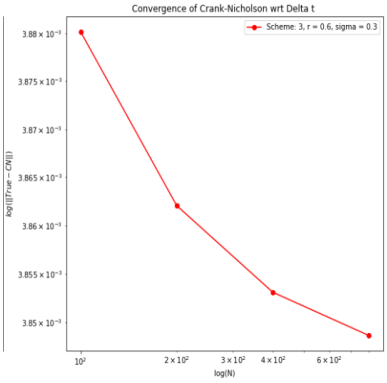


Figure 6: C-N Scheme 3

5.732×10^{-5} to 5.748×10^{-5}. The result of scheme 3 ranges from 4.482×10^{-4} to 4.494×10^{-4}.

It is noticed from scheme 1 and 2, that when we fix the rate of return r, as volatility increases, the results are more accurate. From scheme 1 and 3, when we fix the volatility, we will also get more accurate answers while ratio of return goes larger.

When considering the scheme 2 and 3, since the calculation will take a relatively long time, we can trade off accuracy for a shorter time by using Monte Carlo methods. The run time table of three methods with parameters shown in 5.1:

	300 Iterations	1 Iteration	Full time steps	1 time step
Backward	N/A	N/A	170 ms	N/A
SOR	1 min 50 s	0.36 s	360 ms	N/A
C-N	144 mins	17.3 s	N/A	N/A
Monte Carlo	N/A	N/A	5.351 s	14.5 ms

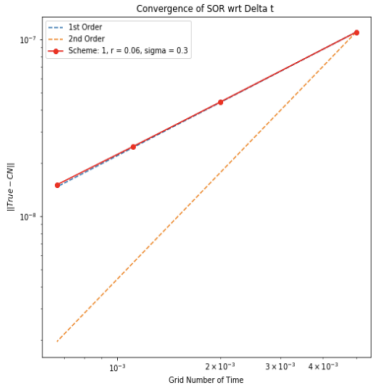


Figure 7: SOR Scheme 1 Convergence

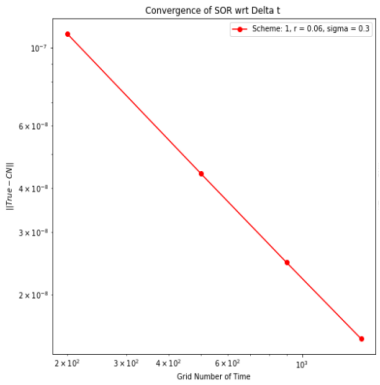


Figure 8: SOR Scheme 1

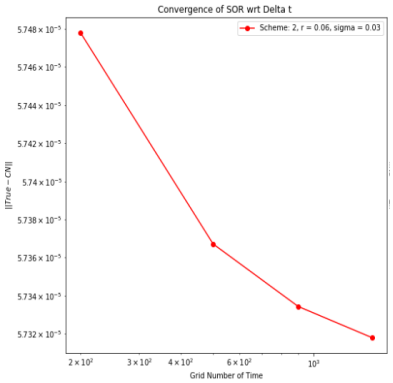


Figure 9: SOR Scheme 2

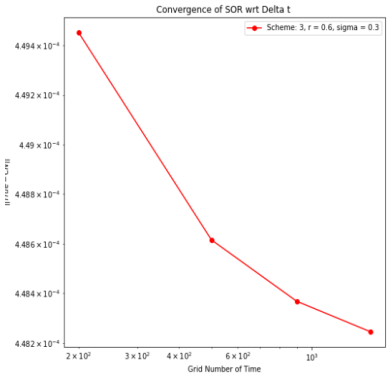


Figure 10: SOR Scheme 3

In order to compare the convergence of four different methods, we plotted the graphs in 11:

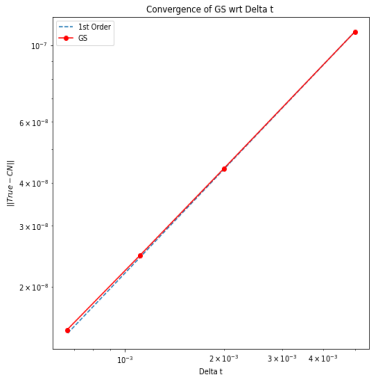


Figure 11: Backward Euler Convergence

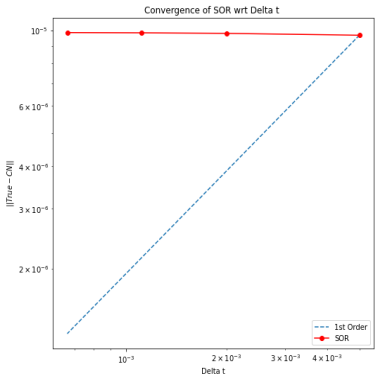


Figure 12: SOR Convergence

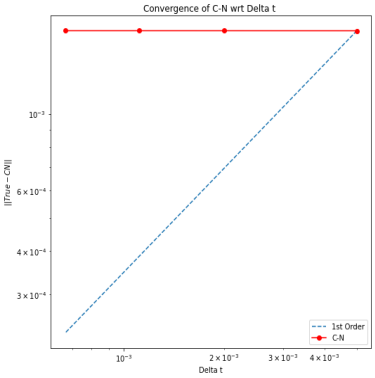


Figure 13: C-N Convergence

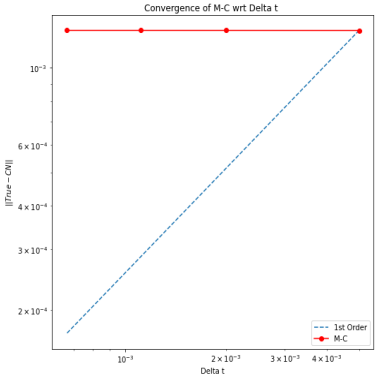


Figure 14: M-C Convergence

We can see that the SOR, Backwards Euler converges as expected, in Figures [7, 11]. We can observe that SOR converges faster than the Backwards Euler method. When choosing the same scheme, it can be noticed that the Crank-Nicholson converges faster than SOR, which is as expected based on their stability analysis.

We can actually seen that the backward Euler is convergent to the first order. Since it is known that Backward Euler method will convergent to the first order, then we can know the calculation of our convergence part is right. However, we can see all other three did not converge to a specific order.

Whether other three methods will converge or not mainly based on the the grid numbers we choose and the iteration numbers, for the current scheme and that specific set of parameters, even though we can see the last three methods actually convergent, they did not show that pattern. In order to know more about the behavior of the convergence of the last three methods, a comparison

taking N_space, N_time, and number of iteration into consideration should be conducted. Since multiple parameters of Black-Scholes also have impact on the behaviors of the numerical methods, we then need to show the influence of them in the further research.

References

[1] Jesus Perez Colino. Least squares monte carlo and the american option.

[2] R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations Steady State and Time Dependent Problems.*

[3] Kyle Mandli. Class notes apma 4301.

[4] Lars Stentoft. Convergence of the least squares monte carlo approach to to american option valuation, Sep 2004.

[5] Tomas Sundvall and David Trang. Examination of impact from different boundary conditions on the 2d black-scholes model. *Uppsala Universitet*, 4, Aug 2014.

[6] Quintus Zhang. Gridplot.py, May 2017.