# Monte Carlo and Quasi Monte Carlo for Option Pricing

with the implementation of Pseudo, Antithetic and Quasi Random Sequences

Hang Miao

## 1 Introduction

The increase in the complexity of derivative securities in recent years accompanied by the constant growth of computational power over time, has led to a great demand of Monte Carlo simulation method for valuation of security, estimation of sensitivity, risk analysis, and stress test portfolio etc. The popularity is due to its direct and simple implementation and robust result. The robustness is guaranteed by law of large number (LLN): The Monte Carlo integration always converge to the true integral as sample size goes to infinity. The robustness comes with the cost of its decelerated converging rate. By central limit theorem (CLT), the converging rate is $\mathcal{O}\left(n^{-\frac{1}{2}}\right)$. Such decelerated converging rate indicates that additional 4 factor of increase in computational effort only result in 2 factors of accuracy improvement. Therefore, even the moderate improvement of convergence rate can have significant impact on the efficiency and range of applicability of Monte Carlo method.

The aim of this paper is to discuss and implement methods that significantly enhance the computational speed of Monte Carlo simulation. At the core of nearly all Monte Carlo simulation is the sequence of random numbers used to drive the simulation. Generating true random numbers is costly and time-consuming. For example, online poker site rely on unpredictable processes like thermal or atmospheric noise to generate true random numbers to prevent reverse engineering the algorithm of card dealing. This level of randomness is not necessary, even harmful for our purpose of pricing securities. Since random generator is often called thousands or even millions of times in a single simulation for pricing the simplest derivatives, the speed of generator is crucial to the survival of Monte Carlo method. Moreover, genuinely random sequence is hard to reproduce easily, while in finance applications, it is often important to use the exact same input in two or more different simulations.

The demand for random generator with fast speed and reproducibility is easily accomplished by using linear congruential generator, proposed by Lehmer (1951), takes the form

$$\begin{aligned} x_{i+1} &= (ax_i + c) \bmod m \\ u_{i+1} &= x_{i+1}/m \end{aligned}$$

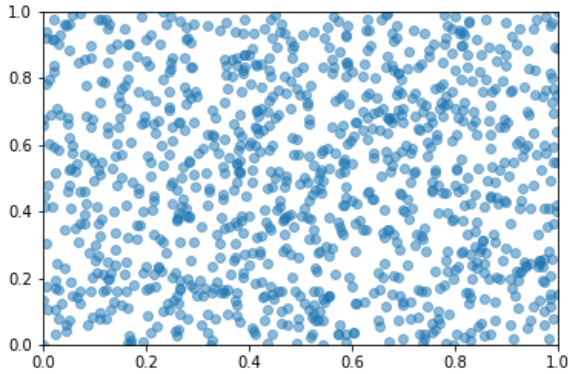
where $x_0$ is the seed, $\{u_i\}_{i=1}^n$ is the pseudo random sequence. The pseudo random number is produced by completely deterministic algorithm. Nonetheless, it is sufficiently good at mimicking

genuine random sequence in the sense that $\{u_i\}_{i=1}^n$ are (i) uniformly distributed between 0 and 1; (ii) mutually independent. It is hard to imagine any other algorithm simpler and faster than the linear congruential generator. Based on the choice of parameters $a$ and $m$ ($c = 0$ normally for algorithm speed consideration), there are various different implementation of linear congruential generator: Park and Miller (1988) has the strongest overall performance and is the most widely used algorithm, thus it will be implemented in this paper; other algorithms like Fishman and Moore (1986) having slightly better uniformity and L'ecuyer (1988) having an advantage on computation could be implemented in the same way with different $a$ and $m$.

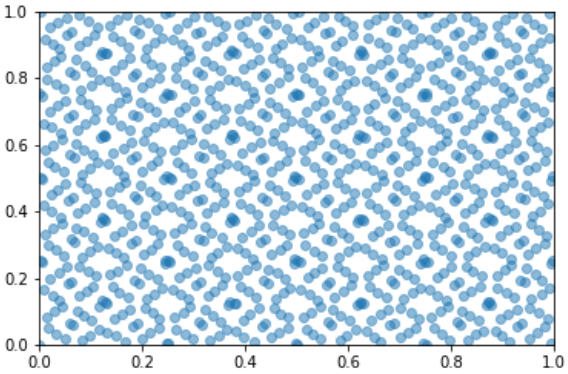
An alternative approach to accelerate the computational speed is variance reduction. Much of the development in Monte Carlo has been in constructions of variance reduction methods which reduces the constant in front of $\mathcal{O}\left(n^{-\frac{1}{2}}\right)$ for Monte Carlo method. There are a variety of variance reduction method, while antithetic variates method is implemented in this article due to its generic applicability that improve the efficiency for any type of simulation problem. Antithetic variates methods attempts to reduce the variance by introducing negative dependence between pairs of replications. It works as follows, given a sequence of paths $\{Y_i\}_{i=0}^n$, a parallel paths $\{\widehat{Y}_i\}_{i=0}^n$ could be generated by inverse CDF transformation $F^{-1}\big(1-F(Y_i)\big)$. Thus both $\{Y_i\}_{i=0}^n$ and $\{\widehat{Y}_i\}_{i=0}^n$ have the same distribution $F$ but are antithetic to each other. For distribution symmetric about origin, the antithetic pairs have same magnitude but opposite signs. Within an antithetic pair, one large value is always accompanied by a small value. This suggest that any path with extreme values will always be balanced by the antithetic paths, resulting in a reduction in variance.

Another method that significantly improve the computational efficiency is to change the choice of random sequence. Instead of using pseudo random sequence to imitate the behavior of the genuine random sequences, quasi Monte Carlo uses a deterministic sequence which is chosen to be more evenly dispersed throughout the region of integration than random sequences as showed in fig. 1. The deterministic sequences with this property are known as low-discrepancy sequences or quasi-random sequences. Discrepancy measures the extent to which the points are evenly dispersed throughout a region: the more evenly dispersed the points are the lower the discrepancy. Low-discrepancy sequences have the property that as successive points are added the entire sequence of points still remains more or less evenly dispersed throughout the region. This property also holds for multi-dimensional integrals approximations as well. For the same reason, quasi Monte Carlo using low discrepancy sequences converges more rapidly at a rate $\mathcal{O}\big(n^{-1}\log(N)^k\big)$

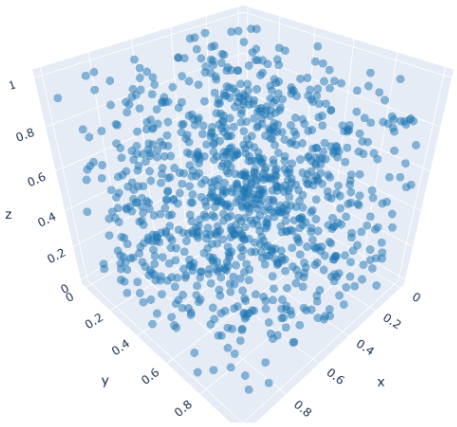
The rest of paper is structured as follows: section 2 set up the Black-Scholes environment, where underlying asset process follows geometric Brownian motion; lay out the foundation for the general derivative pricing formula under risk neutral measure using noarbitrage argument; Payoff function for Asian call option and European call option is provided at the end. Section 3 discuss in detail of three different methods for pricing Asian call option and their implementation algorithm in C++. Section 3.1 is about Monte Carlo with pseudo random sequence generated by Park Miller algorithm. Section 3.2 is about variance reduction method using antithetic random sequence. Section 3.3 is about quasi Monte Carlo using Sobol low discrepancy sequences. Section 4 follows the same structure as section 3 except pricing the European option. Section 5 evaluate different methods under different circumstances based on testing result.



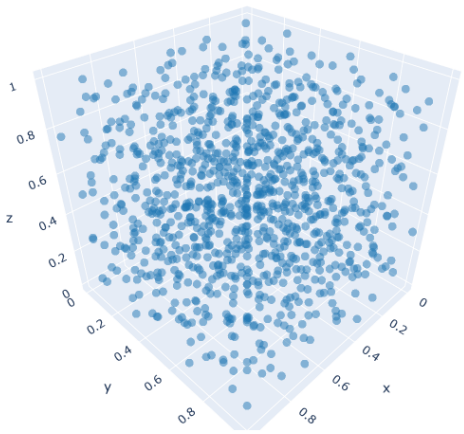

(a) 2D Park Miller Pseudo Random Number




(b) 2D Sobol Quasi Random Number




(c) 3D Park Miller Pseudo Random Number




(d) 3D Sobol Quasi Random Number

Figure 1: Comparison between Psuedo and Quasi Random Sequences. Each sub-figure contains 1024 of 2d-tuples or 3d-tuples generated by Park Miller or Sobol sequence

# 2 Model Setting

## 2.1 Underlying Stock Price Process under Risk Neutral Measure

Suppose the underlying stock price follows Geometric Brownian with continuous dividend payment d.

$$\mathrm{d}S_t = (\alpha - d)S_t\mathrm{d}t \ + \ \sigma S_t \mathrm{d}W_t$$

The market money account risk free return is $r$. Let the market price of risk $\theta = \frac{\alpha - r}{\sigma}$, and apply Girsanov's Theorem to change the physical measure to risk neutral measure under which $\mathrm{d}\widetilde{W}_t = \theta \mathrm{d}t + \mathrm{d}W_t$ is a Brownian Motion. The Stock price under risk neutral measure is

$$\mathrm{d}S_t = (r - d)S_t\mathrm{d}t \ + \ \sigma S_t \mathrm{d}\widetilde{W}_t$$

The discounted portfolio of holding stock and money account is a martingale under risk neutral measure. Given any derivatives which payoff is $V(T)$ where $V(T)$ is $\mathcal{F}_T$-measurable. Applying

martingale representation Theorem, we can find an adapted process $\Gamma(t)$ such that the short position of this derivative can be hedged. Under the no arbitrage condition, the price of the derivative should be the same as the value of the portfolio hedge the short position. Therefore,

$$D_t V_t = D_t X_t = \widetilde{\mathbb{E}}[\, D_T X_T \,|\, \mathcal{F}_t] = \widetilde{\mathbb{E}}[\, D_T V_T \,|\, \mathcal{F}_t] \tag{1}$$

Since $r$ is constant, $D_t = \mathrm{e}^{-rt}$ is a deterministic discount factor independent of $\mathcal{F}_t$.

### 2.2 Payoff Function for Arithmetic Asian Call Option

The payoff function for arithmetic Asian call option is:

$$V_T \;=\; \left(\frac{1}{d}\sum_{k=1}^{d} S_{t_k} - K\right)^+ \tag{2}$$

Where $0 = t_0 < t_1 < \cdots < t_d = T$ are discretely sampled dates

### 2.3 Payoff Function for European Vanilla Call Option

The payoff function for European Vanilla call option is:

$$V_T \;=\; (S_T - K)^+ \tag{3}$$

## 3 Pricing the Arithmetic Asian Call Option

### 3.1 Monte Carlo Simulation

There is no closed form solution for Arithmetic Asian Option. The Monte Carlo simulation method is used for compute the discounted expected payoff. According to the strong law of large number. The average of a sequence of i.i.d random variable converge almost surely to the expected value. So we can estimate

$$e^{-rT}\widetilde{\mathbb{E}}\left[\left(\frac{1}{d}\sum_{k=0}^{d-1} S_{t_k}\mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)h+\sigma\sqrt{h}\mathbf{Z}} - K\right)^+\right]$$

by simulate a large number of underlying equity path using Euler scheme. The Euler Scheme for the approximation of SDE for log spot is

$$\log S^{(i)}_{t_{k+1}} = \log S^{(i)}_{t_k} + (r - d - \frac{1}{2}\sigma^2)h + \sigma\sqrt{h}\mathbf{Z}^{(i)}_{t_k},\ k = 0, \cdots 11$$

where $\log S_{t_0} = \log(100),\ h = \frac{1}{12},\ \mathbf{Z}^{(i)}_{t_k} \sim \text{Normal}(0,1)$. Then the payoff for each path could be computed by exponentiate each log spot and take the sample mean. The risk neutral formula could then be approximated by the sample mean payoffs of each simulated path. $\mathbf{Z}^{(i)}_{t_k}$ for each subinterval in Euler Scheme is generated by Park Miller linear congruential random generator implemented by "random2" and "ParkMiller". This Monte Carlo option pricing algorithm is implemented in the files "ExoticBSEngine" and "PathDependentAsian". I write a statistc getherer in "MCSamples" in order to record the simulation result and compute the standard error( i.e standard deviation for sample mean).

## 3.2 Monte Carlo Simulation with Antithetic Variates

The pair $(Y, \widetilde{Y})$ is antithetic if $Y = g(Z)$ and $\widetilde{Y} = g(Z)$, where $Z$ is a standard normal random variable and $g(z)$ a deterministic function. Using the antithetic pair could enhance the simulation efficiency and possibly reduce the standard deviation of the Monte Carlo sample mean estimator. The underlying log equity price process is simulated by Euler Scheme:

$$\log S_{t_{k+1}}^{(i)} = \log S_{t_k}^{(i)} + (r - d - \frac{1}{2}\sigma^2)h + \sigma\sqrt{h}\mathbf{Z}_{t_k}^{(i)},\ k = 0, \cdots 11$$

$$\log \widetilde{S}_{t_{k+1}}^{(i')} = \log \widetilde{S}_{t_k}^{(i')} + (r - d - \frac{1}{2}\sigma^2)h - \sigma\sqrt{h}\mathbf{Z}_{t_k}^{(i)},\ k = 0, \cdots 11$$

where $\log S_{t_0}^{(i)} = \log \widetilde{S}_{t_0}^{(i')} = \log(100),\ h = \frac{1}{12},\ \mathbf{Z}_t^{(i)} \sim \text{Normal}(0, 1)$. Let

$$\begin{aligned} Y_i &= \mathrm{e}^{-rT}\left(\frac{1}{d}\sum_{k=1}^{d} S_{t_k}^{(i)} - K\right)^+ \\ \widetilde{Y}_i &= \mathrm{e}^{-rT}\left(\frac{1}{d}\sum_{k=1}^{d} \widetilde{S}_{t_k}^{(i')} - K\right)^+ \end{aligned}$$

As a result the antithetic Monte Carlo estimator is:

$$\widehat{Y}_{\text{AV}} := \frac{2}{N}\sum_{i=1}^{\frac{N}{2}}\left(\frac{Y_i + \widetilde{Y}_i}{2}\right), \quad \text{where } N \text{ is the number of path.}$$

Note that number of $\mathbf{Z}_{t_k}^{(i)}$ generated is just the half of pure Monte Carlo does since the diffusion part for path $i'$ is just the negative of that in path $i$. The antithetic variates is implemented in "AntiThetic".

## 3.3 Quasi Monte Carlo Simulation with Sobol Sequence

The concept that low-discrepancy sequences provide an efficient method of "filling" the hypercube $[0, 1]^d$ with a smaller number points for quasi-Monte Carlo integration than might be required by traditional numerical integration for the same accuracy.

In contrast with the regular Monte Carlo method, in which, $\mathbf{Z}^{(i)}$ is obtained by first generating a uniform random number using Park Miller method, then convert that uniform random number to $\mathbf{Z}^{(i)}$ by the standard normal inverse CDF function, Quasi Monte Carlo produce the $\mathbf{Z}^{(i)}$ by first obtain instead of uniform random number but a 1-dimensional low discrepancy number and then convert this number to $\mathbf{Z}^{(i)}$ by the same standard normal inverse CDF function. A quasirandom or low discrepancy sequence, such as the Faure, Halton, Hammersley, Niederreiter or Sobol sequences, is "less random" than a pseudorandom number sequence, but more useful for such tasks as approximation of integrals in higher dimensions, and in global optimization. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers. Algorithms that use such sequences may have superior convergence.

Preponderance of the experimental evidence amassed to date suggest that Sobol sequences are

in many aspects superior to other low discrepancy sequences in financial application. Therefore, I use Sobol sequence to replace the previous pseudo random sequence produced by Park Miller. Considering there are a lot of Sobol generators works well and effective but its interface is not what the rest of code expect, I use the adapter design pattern to create an adapter class that fits the interface and having the existing effective Sobol generator as inner object. The adapter class is implemented in "Random_Sobol". The inner object of Sobol sequence is implemented in "sobol".

# 4 Pricing the European Style Vanilla Call Option

## 4.1 Closed Form Formula

The closed form solution for the European Style vanilla option is derived as follow:
The underlying stock price SDE under risk neutral measure is:

$$\mathrm{d}S_t = (r-d)S_t\mathrm{d}t \ + \ \sigma S_t \mathrm{d}\widetilde{W_t}$$

Since the stock price process follows geometric Brownian Motion, we get the solution for the above SDE as:

$$S_t = S_0 \mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)t+\sigma\widetilde{W}_t} = S_0\mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}\mathbf{Z}}, \quad \text{where } \mathbf{Z} \sim \text{Normal}(0,1)$$

Substitute into the risk neutral pricing formula above, we have

$$V_0 = \mathrm{e}^{-rt}\widetilde{\mathbb{E}}[(S_0\mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}\mathbf{Z}} - K)^+]$$

$$= \mathrm{e}^{-rt}\int_{-\infty}^{d_-}(S_0\mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}z} - K)\frac{1}{2\pi}\mathrm{e}^{-\frac{z^2}{2}}\mathrm{d}z$$

The calculation of the integral lead us to the formula for the call option

$$V_0(t) = \mathrm{e}^{-rt}[FN(d_+) - KN(d_-)]$$

where $F = S_0\mathrm{e}^{(r-d)t}$, $d_\pm = \frac{1}{\sigma\sqrt{t}}\left[\log\frac{S_0}{K} + (r-d\pm\frac{1}{2}\sigma^2)t\right]$

plug into all the given parameters and use the "CumulativeNormal(double x)" function in the "Normals.h" to compute $N(\pm d_\pm)$ we have the option price calculated via explicit formula written in file "BSMformula".

## 4.2 Monte Carlo Simulation

According to the strong law of large number. The average of a sequence of random variable converge almost surely to the expected value. So we can estimate

$$e^{-rT}\widetilde{\mathbb{E}}\left[(S_0\mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)T+\sigma\sqrt{T}\mathbf{Z}} - K)^+\right]$$

by simulate a large number of underlying equity pathes following $S_0\mathrm{e}^{(r-d-\frac{1}{2}\sigma^2)t+\sigma\sqrt{t}\mathbf{Z}}$ using Euler scheme. The Euler Scheme for the approximation of SDE for log spot is

$$\log S_T^{(i)} = \log S_0^{(i)} + (r-d-\frac{1}{2}\sigma^2)h + \sigma\sqrt{h}\mathbf{Z}^{(i)}$$

where $\log S_0^{(i)} = \log(100),\ h = \frac{1}{T},\ \mathbf{Z}^{(i)} \sim \text{Normal}(0,1)$. Then the payoff for each path could be computed by exponentiate each log spot. The risk neutral formula could then be approximated by the sample mean payoffs of each simulated path. $\mathbf{Z}^{(i)}$ for each subinterval in Euler Scheme is generated by Park Miller linear congruential random generator implemented by "random2" and "ParkMiller". This Monte Carlo option pricing algorithm is implemented in the files "ExoticBSEngine" and "PathDependentAsian"(Vanilla is a special case of Asian which only take one sampling point at the maturity time). I write a statistc getherer in "MCSamples" in order to record the simulation result and compute the standard error( i.e standard deviation for sample mean).

## 4.3 Monte Carlo Simulation with Antithetic Variates

The pair $(Y, \widetilde{Y})$ is antithetic if $Y = g(Z)$ and $\widetilde{Y} = g(Z)$, where $Z$ is a standard normal random variable and $g(z)$ a deterministic function. Using the antithetic pair could enhance the simulation efficiency and possibly reduce the standard deviation of the Monte Carlo sample mean estimator. The underlying log equity price process is simulated by Euler Scheme:

$$\log S_T^{(i)} = \log S_0^{(i)} + (r - d - \frac{1}{2}\sigma^2)h + \sigma\sqrt{h}\mathbf{Z}^{(i)}$$
$$\log \widetilde{S}_T^{(i')} = \log \widetilde{S}_0^{(i)} + (r - d - \frac{1}{2}\sigma^2)h - \sigma\sqrt{h}\mathbf{Z}^{(i)}$$

where $\log S_0^{(i)} = \log \widetilde{S}_0^{(i')} = \log(100),\ h = \frac{1}{T},\ \mathbf{Z}^{(i)} \sim \text{Normal}(0,1)$. Let

$$\begin{aligned} Y_i &= \mathrm{e}^{-rT}\left(S_T^{(i)} - K\right)^+ \\ \widetilde{Y}_i &= \mathrm{e}^{-rT}\left(\widetilde{S}_T^{(i')} - K\right)^+ \end{aligned}$$

As a result the antithetic Monte Carlo estimator is:

$$\widehat{Y}_{\text{AV}} := \frac{2}{N}\sum_{i=1}^{\frac{N}{2}}\left(\frac{Y_i + \widetilde{Y}_i}{2}\right), \quad \text{where } N \text{ is the number of path.}$$

Note that number of $\mathbf{Z}^{(i)}$ generated is just the half of pure Monte Carlo does since the diffusion part for path $i'$ is just the negative of that in path $i$. The antithetic variates is implemented in "AntiThetic".

## 4.4 Quasi Monte Carlo Simulation with Sobol Sequence

The concept that low-discrepancy sequences provide an efficient method of "filling" the hypercube $[0,1]^d$ with a smaller number points for quasi-Monte Carlo integration than might be required by traditional numerical integration for the same accuracy.
In contrast with the regular Monte Carlo method, in which, $\mathbf{Z}^{(i)}$ is obtained by first generating a uniform random number using Park Miller method, then convert that uniform random number to $\mathbf{Z}^{(i)}$ by the standard normal inverse CDF function, Quasi Monte Carlo produce the $\mathbf{Z}^{(i)}$ by first

obtain instead of uniform random number but a 1-dimensional low discrepancy number and then convert this number to $\mathbf{Z}^{(i)}$ by the same standard normal inverse CDF function. A quasirandom or low discrepancy sequence, such as the Faure, Halton, Hammersley, Niederreiter or Sobol sequences, is "less random" than a pseudorandom number sequence, but more useful for such tasks as approximation of integrals in higher dimensions, and in global optimization. This is because low discrepancy sequences tend to sample space "more uniformly" than random numbers. Algorithms that use such sequences may have superior convergence.
Preponderance of the experimental evidence amassed to date suggest that Sobol sequences are in many aspects superior to other low discrepancy sequences in financial application. Therefore, I use Sobol sequence to replace the previous pseudo random sequence produced by Park Miller. Considering there are a lot of Sobol generators works well and effective but its interface is not what the rest of code expect, I use the adapter design pattern to create an adapter class that fits the interface and having the existing effective Sobol generator as inner object. The adapter class is implemented in "Random_Sobol". The inner object of Sobol sequence is implemented in "sobol".

## 4.5 Conclusion and Discussion

From Table 1 and Table 2 , we see that program result matches with the QuantLib benchmark result in terms of the means of the derivative. However, in terms of the standard deviation, they are quite different except pseudo random sequence using Park Miler algorithm. The cause of difference for antithetic variates is the different number of samples used in our program and QantLib. When generated a sequence of antithetic random sequence, I used the decoration design pattern on original Park Miler random generator: keep the even number same as Park Miler random number ($U^a_{2i} = U_{2i}$), antithesize the odd number of the sequence ($U^a_{2i+1} = 1 - U_{2i}$). In this way, the number of pseudo sequence using Park Miler and the number of antithetic sequence using this decoration class is exactly same. Even though the antithetic sequence is theoretically more spread out than the original undecorated sequence, but their convergence rate is at the same order. While for QuantLib, the antithetic algorithm is implemented in a entirely different way: generating a sequence of normal random varaibles $\{Z_i\}^n$ using pseudo random generator and Marsaglia-Bray-Box-Muller algorithm, then antithesize the sequence by $Z^a_i = -Z_i$, the final random sample path are the original pseudo sequence plus antithetical sequence, the number doubled. This leads to the $\mathcal{O}\left(\frac{1}{\sqrt{2}}\right)$ difference in standard deviation of antithetical variates.

The standard deviation for quasi Monte Carlo is invalid. Since low discrepancy sequence used in simulation is not statistically mutually independent, rather they followed determined pattern which make them evenly dispersed in the given region as showed in Figure 1. Such property increase the efficiency of Monte Carlo integration. The result from both our program Table 1 and QuantLib benchmark Table 2 conform with our intuition. This could be easily verified for vanilla option, since the its closed form solution is available. The estimated mean using low discrepancy sequence outperform the rest using pseudo or antithetic sequence.

In terms of computational speed, pseudo random sequence by Park Miler achieves the fastest speed due to its simplicity of linear congruential generator. Antithetic algorithm shares the same order of speed as pseudo generator, it normally takes a bit of extra time for antithesize the sequence.

Sobol sequence takes the longest time due to its complexity of generation process.

# 5 Results and Conclusion

## 5.1 Program Result and Benchmark Result

Table 1: C++ implementation result for Asian and Vanilla Call Option. Spot price $S_0 = 100$, risk free interest rate $r = 0.05$, annualized volatility $\sigma = 0.3$, expire in one year $T = 1$, number of path $N = 100000$ for MC simulation, and number of observation $n = 12$ for Asian option.

| | | Asian | | | Vanilla | | | |
|---|---|---|---|---|---|---|---|---|
| K | | Pseudo | Antithetic | Quasi | Pseudo | Antithetic | Quasi | Closed |
| 80 | mean | 22.2181 | 22.2134 | 22.242 | 26.4697 | 26.438 | 26.4603 | 26.4621 |
| | sd | 0.0550287 | 0.0548172 | 0.0550354 | 0.0874619 | 0.0875912 | 0.0875633 | na |
| | speed | 0.106 | 0.159 | 0.117 | 0.018 | 0.034 | 0.044 | 0.001 |
| 90 | mean | 14.3983 | 14.3876 | 14.4212 | 19.6985 | 19.6537 | 19.6957 | 19.6975 |
| | sd | 0.0493339 | 0.0491221 | 0.0489242 | 0.0800243 | 0.0802167 | 0.0801199 | na |
| | speed | 0.106 | 0.105 | 0.114 | 0.019 | 0.024 | 0.043 | 0.001 |
| 100 | mean | 8.44921 | 8.44313 | 8.47532 | 14.2117 | 14.1743 | 14.2296 | 14.2312 |
| | sd | 0.0406482 | 0.0403888 | 0.0406165 | 0.0711287 | 0.0713543 | 0.0711843 | na |
| | speed | 0.111 | 0.142 | 0.131 | 0.018 | 0.033 | 0.047 | 0.001 |
| 110 | mean | 4.51997 | 4.50814 | 4.53842 | 9.99851 | 9.97437 | 10.0185 | 10.0201 |
| | sd | 0.0309603 | 0.0306443 | 0.0308892 | 0.0616101 | 0.0618688 | 0.0616492 | na |
| | speed | 0.124 | 0.174 | 0.128 | 0.02 | 0.024 | 0.042 | 0.001 |
| 120 | mean | 2.2386 | 2.22119 | 2.24466 | 6.88889 | 6.88087 | 6.90249 | 6.90401 |
| | sd | 0.022172 | 0.0217871 | 0.0220509 | 0.0522503 | 0.0525521 | 0.0522884 | na |
| | speed | 0.141 | 0.105 | 0.125 | 0.023 | 0.024 | 0.049 | 0.001 |
| 150 | mean | 0.202261 | 0.188019 | 0.196561 | 2.051 | 2.06489 | 2.05667 | 1.77526 |
| | sd | 0.00664198 | 0.00619794 | 0.00633633 | 0.0293928 | 0.0298233 | 0.0293631 | na |
| | speed | 0.099 | 0.097 | 0.11 | 0.018 | 0.024 | 0.043 | 0.001 |

Table 2: QuantLib Benchmark Result for Asian and Vanilla Call Option. Spot price $S_0 = 100$, risk free interest rate $r = 0.05$, annualized volatility $\sigma = 0.3$, expire in one year $T = 1$, number of path $N = 100000$ for MC simulation, and number of observation $n = 12$ for Asian option.

| | | Asian | | | Vanilla | | | |
|---|---|---|---|---|---|---|---|---|
| K | | Pseudo | Antithetic | Quasi | Pseudo | Antithetic | Quasi | Closed |
| 80 | mean | 22.351157 | 22.233934 | 22.237149 | 26.331299 | 26.424729 | 26.460322 | 26.462086 |
| | sd | 0.055125 | 0.013560 | na | 0.087269 | 0.032452 | na | na |
| 90 | mean | 14.413328 | 14.370823 | 14.414539 | 19.695539 | 19.714606 | 19.695740 | 19.697442 |
| | sd | 0.049354 | 0.018380 | na | 0.080464 | 0.037074 | na | na |
| 100 | mean | 8.429718 | 8.489271 | 8.467323 | 14.197152 | 14.260336 | 14.229618 | 14.231255 |
| | sd | 0.040657 | 0.021650 | na | 0.071178 | 0.039305 | na | na |
| 110 | mean | 4.559674 | 4.570664 | 4.536692 | 10.091656 | 10.008264 | 10.018505 | 10.020078 |
| | sd | 0.031064 | 0.019539 | na | 0.062076 | 0.037525 | na | na |
| 120 | mean | 2.230359 | 2.240448 | 2.244534 | 6.942344 | 6.919959 | 6.902489 | 6.903998 |
| | sd | 0.021874 | 0.014802 | na | 0.052719 | 0.033804 | na | na |
| 150 | mean | 0.193475 | 0.193344 | 0.196643 | 2.075034 | 2.083101 | 2.056668 | 2.057986 |
| | sd | 0.006502 | 0.004446 | na | 0.029475 | 0.020533 | na | na |

## 5.2 Convergence Diagnostics

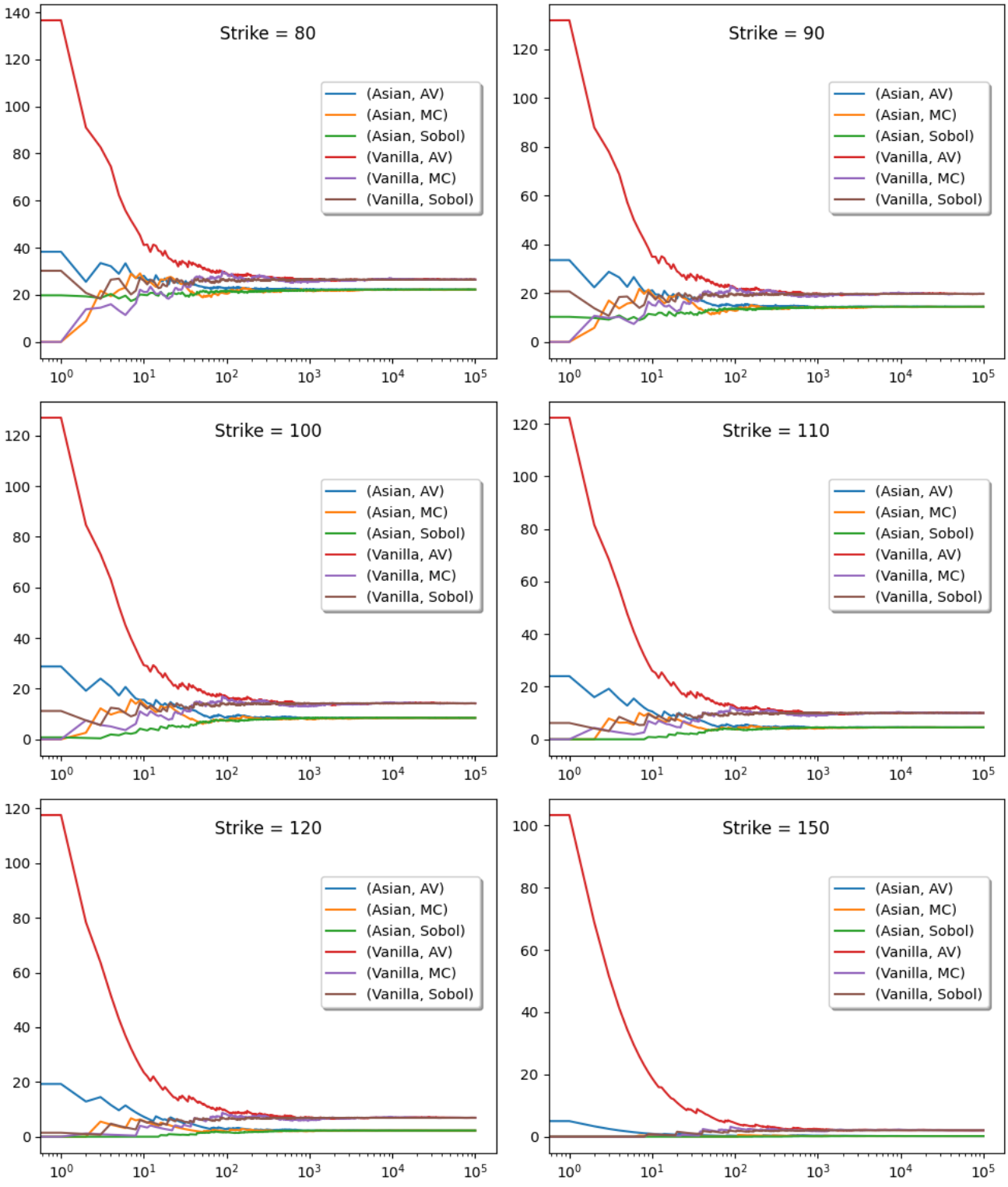



Figure 2: Convergence Diagnostics for Mean

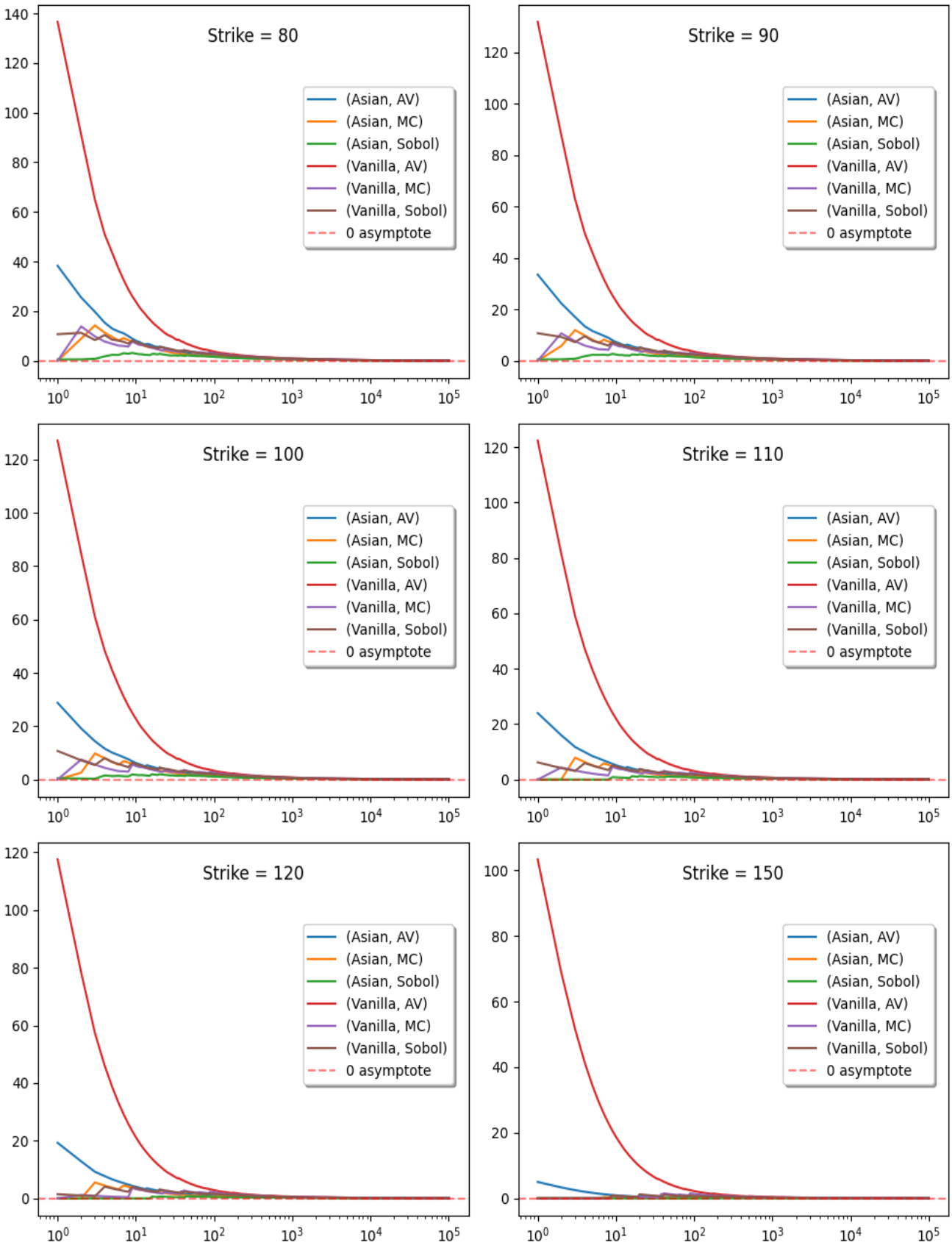



Figure 3: Convergence Diagnostics for Standard Deviation

# References


Boyle, P., Broadie, M., and Glasserman, P. (1997). Monte carlo methods for security pricing. Journal of economic dynamics and control, 21(8-9):1267–1321.

Boyle, P. P. (1977). Options: A monte carlo approach. Journal of Financial Economics, 4(3):323–338.

Broadie, M. and Glasserman, P. (1996). Estimating security price derivatives using simulation. Management science, 42(2):269–285.

Curran, M. (1994). Valuing asian and portfolio options by conditioning on the geometric mean price. Management science, 40(12):1705–1711.

Fishman, G. S. and Moore, III, L. R. (1986). An exhaustive analysis of multiplicative congruential random number generators with modulus 2ˆ31-1. SIAM Journal on Scientific and Statistical Computing, 7(1):24–45.

Fusai, G. and Kyriakou, I. (2016). General optimized lower and upper bounds for discrete and continuous arithmetic asian options. Mathematics of Operations Research, 41(2):531–559.

Geman, H. and Yor, M. (1993). Bessel processes, asian options, and perpetuities. Mathematical finance, 3(4):349–375.

Glasserman, P. (2004). Monte Carlo methods in financial engineering, volume 53. Springer.

Harrison, J. and Kreps, D. M. (1979). Martingales and arbitrage in multiperiod securities markets. Journal of Economic Theory, 20(3):381–408.

Harrison, J. and Pliska, S. R. (1981). Martingales and stochastic integrals in the theory of continuous trading. Stochastic Processes and their Applications, 11(3):215–260.

Horvath, A. and Medvegyev, P. (2016). Pricing asian options: a comparison of numerical and simulation approaches twenty years later. Journal of Mathematical Finance, 6:810–841.

Joshi, M. S. and Joshi, M. S. (2004). C++ design patterns and derivatives pricing, volume 1. Cambridge University Press.

Kemna, A. G. and Vorst, A. C. (1990). A pricing method for options based on average asset values. Journal of Banking & Finance, 14(1):113–129.

L'ecuyer, P. (1988). Efficient and portable combined random number generators. Communications of the ACM, 31(6):742–751.

Lehmer, D. H. (1951). Mathematical methods in large-scale computing units. Annu. Comput. Lab. Harvard Univ., 26:141–146.

Park, S. K. and Miller, K. W. (1988). Random number generators: good ones are hard to find. Communications of the ACM, 31(10):1192–1201.

Rubinstein, M. and Reiner, E. (1991). Breaking down the barriers. 4(8):28–35.

Shreve, S. E. et al. (2004). Stochastic calculus for finance II: Continuous-time models, volume 11. Springer.

Thompson, G. W. P. (1999). Fast narrow bounds on the value of asian options. Technical report.

Turnbull, S. M. and Wakeman, L. M. (1991). A quick algorithm for pricing european average options. Journal of financial and quantitative analysis, 26(3):377–389.