An Efficient Quasi-Monte Carlo Simulation for Pricing Asian Options under Heston's Model

by

Kewei Yu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Quantitative Finance
in
Quantitative Finance

Waterloo, Ontario, Canada, 2015

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The market for path-dependent options has been expanded considerably in the financial industry. The approach for pricing the path-dependent options in this thesis is developed by Kolkiewicz (2014) based on a quasi-Monte Carlo simulation with Brownian bridges conditioning on both their terminal values and the integrals along the paths. The main contribution of this essay is an extension of the above method to price Asian options under a stochastic volatility model. A Matlab implementation of generating multi-dimensional independent Brownian paths is also included as part of the contribution. The result can be used to price path-dependent options, such as an Asian option under both stochastic interest rate model and/or stochastic volatility model. A comparison with regular Monte Carlo simulation is provided.

Acknowledgements

I would like to thank all the little people who made this possible.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Pricing path-dependent Options

Nowadays, the trading volumes of path-dependent option are high and most deals take place in OTC market due to several of its characteristics. Firstly, the path-dependent options can provide investors the products specific to their needs. Secondly, the options can play an important role in hedging as they meet hedgers' needs in a cost effective ways. For example, Asian options are less expensive than European options. Hedging strategies based on these options are usually more preferable.

The pricing methods used for main types of path-dependent options are numerical or analytical. The most common approaches include the partial differential equation (PDE) approach and the Monte Carlo approach. This chapter justifies the reason for choosing Monte Carlo simulation for pricing.

The PDE approach solves a PDE with given initial and boundary conditions in order to price an option. When additional variables are introduced in the PDE, this approach can be computationally expensive, especially in the case of stochastic interest rate and/or stochastic volatility (Forsyth at al. 1998, Vecer 2001). For example, the Asian option price under stochastic volatility in the PDE is considered as a function of the underlying stock price, time, volatility and the price average.

The literature on simulation in the context of option pricing started with the paper by Boyle (1977). The price of a derivative is given by the expected discounted payoffs. The expectation is taken with respect to a risk neutral probability measure. The Monte Carlo approach has the following steps:

1. Over a given time horizon, simulate a path of the underlying asset under a risk neutral measure;
2. Discount the payoff corresponding to each path at the risk-free rate;
3. Repeat the first two steps for a large number of paths;
4. Take the average of the discounted payoffs over all the paths simulated to obtain the option's value.

Under the Black-Scholes model for the dynamic of the underlying asset, the formula for a random path generated at discrete time-intervals is as follows:

$$S_{t+\Delta_t} = S_t \exp\left[\left(r - \frac{\sigma^2}{2}\right)\Delta_t + \sigma_t\sqrt{\Delta_t}\epsilon\right] \quad t \in \{0, \Delta_t, 2\Delta_t, \dots\}$$

where

S_t is the asset price at time t;

r represents risk free interest rate;

Δ_t represents the constant length of time steps;

σ_t represents the constant volatility of stock prices;

$\{\epsilon_t | \, t = 0, \Delta_t, 2\Delta_t, \dots\}$ is a vector of i.i.d. standard normal random variables.

`

The standard error of the sample mean converges to 0 with a rate of $\frac{1}{\sqrt{N}}$ if N paths are generated independent and randomly, where N is the total number of paths. This result is guaranteed by the central limit theorem.

For random numbers, the rate of convergence $\frac{1}{\sqrt{N}}$ does not depend on the dimensions. In the next chapters we explain that if we replace random numbers with low-discrepancy numbers then a higher rate of convergence can be achieved. The discretization of each path results in a higher dimension of the problem. However, the methods based on simulation can deal with extremely complicated or high-dimensional problems. Another advantage of a simulation approach is that advanced technology have reduced the computational time which makes this method more attractive. Overall, the method is easy to implement, flexible and easy to modify.

The main drawback of simulation is that the low rate of convergence will require a very high number of simulations for accuracy. This can be improved by many existing variance reduction methods such as antithetic variables, control variables, importance sampling and stratified sampling, etc. The method that we consider uses deterministic sequences of numbers instead of pseudo-random numbers. In the literature, they are called "low-discrepancy" sequences. Kolkiewicz (2014) developed a simulation method which efficiently uses the low-discrepancy sequence so that Brownian paths are generated conditionally on their terminal

values and the integrals along the paths. This simulation method is introduced as the BBI method in this thesis, and is extended to price Asian options under a stochastic volatility model.

The rest of the thesis is organized as follows. Chapter 2 provides a literature review of Asian option and quasi-Monte Carlo simulation. In chapter 3, the efficient quasi-Monte Carlo simulation is introduced in detail. In chapter 4, prices of arithmetic Asian options are simulated under the Heston model. The results of the implementation are compared with results from the regular Monte Carlo simulation. Chapter 5 gives an analysis on the comparison, and Chapter 6 concludes the findings.

Chapter 2

Asian Option and Quasi-Monte Carlo Approach for Stochastic Volatility Models

Payoffs of path-dependent options at expiry are based on the past historical underlying asset prices. For example, in the case of Asian options, we need the average of asset prices over certain fixed dates. This chapter is organized as follows. Asian options will be introduced first. This is followed by two popular stochastic volatility models: Hull-White's model (1987) and Heston's model (1998). We will then introduce a quasi-Monte Carlo simulation method, low discrepancy sequences and a Brownian Bridge construction.

2.1 Asian Options and Monte Carlo Simulation

The payoff of a regular Asian option depends on a strike price and the average (arithmetic, geometric or harmonic) value of the underlying asset over a specific time period. For the arithmetic Asian option, the payoff is as follows:

$$\text{Payoff} = \max[\varphi(A_T - K), 0] \tag{1}$$

where

$A_t = \frac{1}{t}\int_0^t S_s\, ds$ for $t > 0$, is the process of an underlying asset average price;

S_t for $t > 0$, is the process of an underlying asset price;

T represents the time at maturity;

`

K is the strike price;

φ equals 1 for a call option and -1 for a put option.

For any process X, we use X_t to represent the instantaneous value of the process at time $t \in [0,T]$. Thus, A_T represents the average of S_t from $t = 0$ to T.

The geometric average options are relatively easy to price as the price is lognormally distributed and one can apply the Black-Scholes model. However, the sum of lognormal random variables is not lognormal and there is no recognized distribution for that. Unlike the geometric Asian options, which can be priced analytically with a risk neutral expectation by using the fact that the average follows a lognormal distribution, the arithmetic average does not have this property. Therefore, Monte Carlo simulations have been used quite often to price arithmetic Asian options. Monte Carlo simulation becomes a good way to price options mainly due to its advantages compared to other methods: firstly, it generates numbers of path of all desired time asset values in a simple and easy to implemented way. For path dependent options, such as Asian option, barrier option and look back option, Monte Carlo gives a simple and flexible solution. Secondly, one can assess the accuracy of the computation by calculating, for example, a 95% confidence interval for μ_S using the following formula

$$\left(\mu_S - \frac{1.96\sigma_S}{\sqrt{N}}, \mu_S + \frac{1.96\sigma_S}{\sqrt{N}}\right) \quad (2)$$

`

based on the sample mean μ_S, the sample standard deviation σ_S and the standard error $\frac{\sigma_S}{\sqrt{N}}$ of the simulation results and assuming that μ_S is normally distributed.

2.2 Stochastic Volatility Models

In real financial markets, volatility usually does not stay constant over time. Stochastic volatility models make pricing of options more realistic. Hull-White model (1987) proposes stochastic volatility and interest rate models. Recall the process and assumption for pricing for the Monte Carlo simulation method introduced in Chapter 1. We use the same notations here with one exception that the volatility σ is not constant, but follows a stochastic process in a risk-neutral world. Based on a large number of numerical experiments, Hull and White decided to propose a model based on the assumption that the stochastic volatility is independent of the asset price. Then, the asset price process and the volatility process satisfy the equations below:

$$dS_t = rS_t dt + \sigma_t S_t dZ_t$$

$$dV_t = \mu_t V_t dt + \delta V_t dW_t$$

where

$V_t = \sigma_t^2$ represents the variance process of S;

μ_t represents the drift of V;

δ represents the volatility of V;

Z_t and W_t are independent Brownian motions.

`

By Ito's formula, we have:

$$d\ln V_t = \frac{1}{V_t} dV_t - \frac{1}{2} \times \frac{1}{{V_t}^2} (dV_t)^2$$

$$= \left(\mu_t - \frac{\delta^2}{2}\right) dt + \delta dW_t.$$

In our simulations we use the Euler discretization of the two processes:

$$S_{t+\Delta_t} = S_t \exp\left[\left(r - \frac{\sigma^2}{2}\right)\Delta_t + \sigma_t\left(Z_{t+\Delta_t} - Z_t\right)\right]$$

$$V_{t+\Delta_t} = V_t \exp\left[\left(\mu_t - \frac{\delta^2}{2}\right) + \delta\left(W_{t+\Delta_t} - W_t\right)\right]$$

Applying Monte Carlo simulation with stochastic volatility, Hull-White model assumes the following form of the drift term:

$$\mu_t = \alpha(\overline{\sigma} - \sigma_t)$$

and therefore, the simple lognormal process for variance could be replaced by a mean-reverting form

$$dV_t = \alpha(\overline{\sigma} - \sigma_t)V_t dt + \delta V_t dW_t$$

where

α represents the speed of mean reversion;

$\overline{\sigma}$ represents the long term volatility mean.

Below we describe the simulation procedure suggested by Hull and White. Let $\{\varepsilon_t \mid t = 0, \Delta_t, 2\Delta_t, \dots\}$ and $\{\epsilon_t \mid t = 0, \Delta_t, 2\Delta_t, \dots\}$ denote two vectors of i.i.d. standard normal random variables.

`

1. Variance at each point in time was generated by the following stochastic process:

$$V_{t+\Delta_t} = V_t \exp\left(\left(\mu_t - \frac{\delta^2}{2}\right)\Delta_t + \delta\varepsilon_t\sqrt{\Delta_t}\right) + \delta\varepsilon_t\sqrt{\Delta_t}$$

2. Use the variance vector to generate the stock prices:

$$S_{t+\Delta_t} = S_t \exp\left[\left(r - \frac{V_t}{2}\right)\Delta_t + \sqrt{V_t}\epsilon_t\sqrt{\Delta_t}\right]$$

The generated paths can be then used to price Asian options.

Using the same notation as in the Hull-White model, the Heston model can be written as:

$$dS_t = rS_t dt + \sigma_t S_t dZ_t$$

$$dV_t = \alpha\left(\overline{\sigma}^2 - V_t\right)dt + \delta\sigma_t dW_t$$

$$dZ_t dW_t = \rho dt$$

where ρ is the correlation between Z_t and W_t.

Given two independent Brownian motions $W^{(1)}$ and $W^{(2)}$, it can be easily verified that for Z defined as

$$dZ = \rho dW^{(1)} + \sqrt{1-\rho^2}dW^{(2)},$$

the correlation between Z and $W^{(1)}$ is ρ.

Note that the process of V_t in Heston model follows the CIR model (Cox at al. 1985). Thus the condition

$$2\alpha\overline{\sigma}^2 \geq \delta^2$$

`

will ensure that V_t remains positive. However, the simulation formula

$$V_{t+\Delta_t} = V_t + \alpha(\overline{\sigma}^2 - V_t)\Delta_t + \delta\sqrt{V_t}(W_{t+\Delta_t} - W_t)$$

can still produce negative values in the cases of a large negative Gaussian increment $(W_{t+\Delta_t} - W_t)$. The absolute value of $V_{t+\Delta_t}$ will be used as proposed by Berkaoul, Bossy and Diop(2008). Therefore, to simulate price paths under the Heston model we will use the following procedure with two independent Brownian motions W_t and Z_t:

1. Variance at each discrete point is generated according to:

$$V_{t+\Delta_t} = \left|V_t + \alpha(\overline{\sigma}^2 - V_t)\Delta_t + \delta\sqrt{V_t}(W_{t+\Delta_t} - W_t)\right|. \quad (3)$$

2. Then the stock prices at the same discrete time points are obtained using

$$S_{t+\Delta_t} = S_t \exp\left[\left(r - \frac{V_t}{2}\right)\Delta_t + \sqrt{V_t}\left(\rho(W_{t+\Delta_t} - W_t) + \sqrt{1-\rho^2}(Z_{t+\Delta_t} - Z_t)\right)\right]. \quad (4)$$

2.3 Quasi-Monte Carlo Simulation

Quasi-Monte Carlo simulation (Niederreiter, 1992) is based on a similar procedure as Monte Carlo simulation, but it uses sequences of quasi-random numbers that have a more uniform behavior. Similar to pseudo-random numbers, quasi-random numbers are generated algorithmically by computer except the latter have the property of being deterministically chosen based on equally distributed sequences in order to cover uniformly the unit hypercube. The improved convergence rate is almost as fast as $\frac{1}{N}$. However, Niederreiter also pointed that the completely

`

deterministic procedure guarantees deterministic error bounds. There is no simple criterion to assess the accuracy of the estimates.

Quasi-random sequences are also called sub-random sequences or low discrepancy sequences. The discrepancy of a point set $P = \{x_i\}_{i=1,\bar{N}} \in [0,1)^s$ is defined as

$$D_N^{(s)}(P) = \sup_{\mathrm{B}} \left| \frac{A(B;N)}{N} - \lambda(B) \right|$$

where:

the supremum is taken over all sets of B of the form

$$B = [0, t_1) \times \ldots \times [0, t_s), 0 \leq t_j \leq 1, j = 1, \ldots, s;$$

$\lambda(B)$ represents the Lebesgue measure of B;

$A(B;N)$ represents the number of $x_i, i = 1, \ldots, N$, contained in B.

Hence, B is a s-dimensional hyper-rectangle, $\lambda(B)$ is the length, area or volume of B, $\frac{A(B;N)}{N}$ is the percentage of x_i in B, and the discrepancy is the largest difference between $\frac{A(B;N)}{N}$ and $\lambda(B)$.

A low-discrepancy sequence is a sequence of s-dimensional points that fill the sample area rather uniformly. Discrepancy of such a sequence is lower than straight pseudo-random sequences. It follows that a infinite sequence $\{x_i\}$ is uniformly distributed if and only if $\lim_{N\to\infty} D_N^{(S)}(x_1, x_2, x_3, \ldots) = 0$. In other words, the

`

discrepancy can be considered as a quantitative measure of uniformity. Keifer (1961) showed that the discrepancy of a random uniform sequence of length N is bounded below by $(2N^{-1}\log\log N)^{\frac{1}{2}}$. And it turns out, however, that deterministic sequences may have smaller discrepancy. A straight forward example can be $\left\{x_i | x_i = \frac{2i-1}{2N}, i = 1, \dots, N\right\}$, which was constructed using the midpoint rule. Niederreiter (1992) suggested that such a sequence can achieve the lowest possible discrepancy $\frac{1}{2N}$. For an s-dimensional sequence of length N, it is believed that the lower bound can achieve $\frac{B_s \log^s N}{N}$, where $B_s > 0$ depends only on s for some low discrepancy sequences (Kuripers and Niederreiter 1974).

The concept of discrepancy can also be used to determine the upper bounds on integration errors. The famous Koksma-Hlawka inequality (Koksma 1943, Hlawka 1961) is stated after the proper definition of the Hardy-Krause variation.

Definition (Hardy-Krause Variation). For a function f on $[0,1)^s$ with $s > 1$, $x = (x_1, x_2, \dots, x_s)$ and $y = (y_1, y_2, \dots, y_s)$, let us define the s-dimensional difference operator $\Delta^{(s)}$as follows:

$$\Delta^{(s)}(f; [x, y]) = \sum_{i_1=0}^{1} \dots \sum_{i_s=0}^{1} (-1)^{\sum_{j=1}^{s} i_j} f(y_1 + i_1(x_1 - y_1), \dots, y_s + i_s(x_s - y_s)).$$

Let $0 = u_1^{(d)} \leq u_2^{(d)} \leq \cdots \leq u_{n_d}^{(d)} = 1, d = 1, \dots, s$ be a partition of [0,1], and let P be the partition of $[0,1]^s$ which is composed as

`

$$P = \left\{ \left[u_{j_1}^{(1)}, u_{j_1+1}^{(1)} \right] \times \ldots \times \left[u_{j_s}^{(s)}, u_{j_s+1}^{(s)} \right], \text{for } j_d = 1, \ldots, n_d \text{ and } d = 1, \ldots, s \right\}.$$

Then the variation of f on $[0,1)^s$ in the sense of Vitali is defined as the supremum over all partitions P of $[0,1)^s$ as follows:

$$V^{(s)}(f; [0,1]^s) = \sup_{\mathrm{P}} \left| \sum_{[x,y] \in P} \Delta^{(s)}(f; [x,y]) \right|.$$

Let $1 \leq d \leq s$, $1 \leq j_1, \ldots, j_d \leq s$ and $V^{(d)}(f; j_1, \ldots, j_d; [0,1]^s)$ denote the vitali variation of the function f restricted to the d-dimensional subspace $(u_1, \ldots, u_s) \in [0,1]^s$ such that $u_k = 1$ for all $k \neq j_i, \ldots, j_d$. Then the variation of f on $[0,1]^s$ in the sense of Hardy and Krause anchored at 1 is given by

$$V(f) = \sum_{d=1}^{s} \sum_{1 \leq j_1 < \cdots < j_d \leq s} V^{(d)}(f; j_1, \ldots, j_d; [0,1]^s).$$

Theorem (Koksma-Hlawka Inequality). For an s-dimensional sequence $x = (x_1, x_2, \ldots, x_s) \in [0,1)^s$ for $i = 1, \ldots, N$ and any function f of bounded variation in the sense of Hardy-Krause we have

$$\left| \frac{\sum_{i=1}^{N} f(x_i)}{N} - \int_{[0,1)^s} f(u) du \right| \leq V(f) D_N^{(s)}(x).$$

Note that the integration error bound given by Koksma-Hlawka inequality is separated into two parts: the smoothness of the function f and the discrepancy of the deterministic nodes. Recall that the discrepancy for a set of N random points and a sequence of N low discrepancy points are $\mathrm{O}\left(N^{-\frac{1}{2}} (\log\log N)^{\frac{1}{2}}\right)$ and $\mathrm{O}(N^{-1} \log^s N)$

`

respectively. Thus, for a fixed dimension s and a sufficiently large N, the use of low discrepancy sequences would increase the efficiency. The advantages of quasi-Monte Carlo simulation is that it results in faster convergence. Therefore, fewer points are needed to achieve the same level of accuracy. With a well-chosen sequence of points, the Quasi-Monte Carlo simulation gives a better estimate with shorter computational time and higher accuracy.

2.4 Use of Low-discrepancy Sequences

Low-discrepancy sequences cover the unit cube as "uniformly" as possible by reducing gaps and clustering of points (Paskov, 1998). For implementationa, there are several well-known multi-dimensional sequences, like Halton (1964), Faure (1982) and Sobol (1967) sequences. A comparison of the low-discrepancy sequences was presented by Galanti and Jung (1997). Their results suggest that all low-discrepancy sequences can be successfully used for low dimensions. Although implementation of the Halton method is much easier, it's performance is typically dominated by the Faure and Sobol methods. For higher dimensions, Sobol sequences and generalized Faure (Tezuka, 1998) method would typically give fast convergence and reliability. As a motivation for dimension reduction, it is worth mentioning that typical low discrepancy sequences, such as Sobol and Halton sequences, have poor uniformity in their high dimensions, which may cause

simulation error. In addition, the rate at which the integration error diminishes depends on the dimension s.

2.5 Brownian Bridge and Quasi-Monte Carlo Simulation

The payoff function of an Asian option depends on the spot prices of the underlying over the whole time period. Section 2.2 provides the price path simulation formulas under some stochastic volatility models. Equation (3) and (4) show that the uncertainty of the paths depends on the two Brownian paths Z and W. Now we explain how low-discrepancy sequences can be used to simulate paths of a stochastic process. According to Koksma-Hlawka inequality, the multiplicative factor $\log^s N$ in the discrepancy bound of low discrepancy sequence suggests that high dimensions would significantly affect the valuation efficiency. The basic idea of quasi-Monte Carlo simulation is to simply replace random points with low discrepancy points. Paskov and Traub (1995) used this idea to price a collaterized mortgage obligation. The problem used a 360-dimensional low-discrepancy sequence as they are using daily prices to evaluate a one year contract. The authors demonstrated that quasi-Monte Carlo leads to more accurate results than standard methods. Paskov (1997) used the concept of effective dimension and argued that the efficiency of an integration method closely relates to the effective dimension of the problem.

Effective dimension is defined using the concept of ANOVA decomposition of function variance (Caflisch, Morokoff and Owen, 1997). For any non-empty index

set $u \subseteq \{1,2,\dots,s\}$, let u^c and $|u|$ denote its complement and cardinality. Considering an s-dimensional sequence x, we denote by x_u the $|u|$-dimensional sequence containing the coordinates of x with indices in u. For a square integrable function $f(x)$ defined on $[0,1]^s$, the ANOVA decomposition of $f(x)$ is the sum of its ANOVA terms over all non-empty subsets of $\{1,2,\dots,s\}$:

$$f(x) = \sum_{u \subseteq \{1,2,\dots,s\}} f_u(x)$$

where

$$f_u(x) = \int_{[0,1]^{s-|u|}} f(x) dx_{u^c} - \sum_{v \subset u} f_v(x) \text{ and}$$

$$f_\emptyset(x) = \int_{[0,1]^s} f(x) d_x.$$

Note that the ANOVA decomposition is orthogonal in that for $u \neq v$

$$\int_{[0,1]^s} f_u(x) f_v(x) dx = 0.$$

As a result, we have the decomposition for the variance of $f(x)$ as the sum of ANOVA terms over all non-empty subsets of $\{1,2,\dots,s\}$:

$$\sigma^2(f) = \sum_{u \subseteq \{1,2,\dots,s\}} \sigma_u^2(f)$$

where

$$\sigma_u^2(f) = \int_{[0,1]^s} f_u^2(x) dx \text{ and}$$

$$\sigma_\emptyset^2(f) = 0.$$

`

The effective dimension of f in the superposition sense is defined as the smallest integer d_s, such that

$$\sum_{|u|\leq d_s} \sigma_u^2(f) \geq p\sigma^2(f)$$

where $p \in (0,1)$ is the proportion, which normally close to 1. The effective dimension of f in the truncation sense is the smallest integer d_t, such that

$$\sum_{u\subseteq\{1,\ldots,d_t\}} \sigma_u^2(f) \geq p\sigma^2(f).$$

The problem of how path generation methods affect the accuracy of quasi-Monte Carlo methods has been studied by using the concept of effective dimension (Wang and Tan 2012 and the references therein). Considering the ANOVA decomposition for right hand side of the Koksma-Hlawka inequality, the simulation error bound can be expressed as

$$\left|\frac{\sum_{i=1}^N f(x_i)}{N} - \int_{[0,1)^s} f(u)du\right| \leq V(f)D_N^{(s)}(x)$$

$$\leq \sum_{u\subseteq\{1,\ldots,s\}} V_u(f)D_N^{(s)}(P_{x,u})$$

where $P_{x,u}$ is the projection of the sequence x on $[0,1]^{|u|}$. Let d_s and d_t represent the effective dimension of f in the sense of superposition and truncation respectively. We have

$$\left|\frac{\sum_{i=1}^N f(x_i)}{N} - \int_{[0,1)^s} f(u)du\right| \leq \sum_{|u|\leq d_s} V_u(f)D_N^{(s)}(P_{x,u}) + \sum_{|u|>d_s} V_u(f)D_N^{(s)}(P_{x,u}) \text{ and}$$

`

$$\left|\frac{\sum_{i=1}^{N} f(x_i)}{N} - \int_{[0,1)^s} f(u)du\right| \leq \sum_{u\subseteq\{1,\ldots,d_t\}} V_u(f) D_N^{(s)}(P_{x,u}) + \sum_{u\subseteq\{d_t+1,\ldots,s\}} V_u(f) D_N^{(s)}(P_{x,u}).$$

If d_s and d_t are small, the terms $\sum_{|u|\leq d_s} V_u(f) D_N^{(s)}(P_{x,u})$ and $\sum_{u\subseteq\{1,\ldots,d_t\}} V_u(f) D_N^{(s)}(P_{x,u})$ are much smaller for quasi-Monte Carlo methods than for crude Monte Carlo methods. The definition of an effective dimension implies that the terms $\sum_{|u|>d_s} V_u(f) D_N^{(s)}(P_{x,u})$ and $\sum_{u\subseteq\{d_t+1,\ldots,s\}} V_u(f) D_N^{(s)}(P_{x,u})$ are small as the multiplicative factor $V_u(f)$ is small. Thus, quasi-Monte Carlo methods are expected to be more efficient than crude Monte Carlo methods. Note that the argument was made on error bounds. Therefore, small effective dimension is not a sufficient condition for the effectiveness of applying quasi-Monte Carlo methods.

Wang and Tan (2012) investigated the effect of several path generation methods on Asian option pricing. The path generation methods included in the investigation include brownian bridge (Caflisch and Moskovitz, 1995), principal component analysis (Acworth, Broadie, Glasserman, et al., 1996), linear transformation (Imai and Tan, 2006) and diagonal method (Morokoff, 1998). It is concluded that all these four methods results in significantly smaller effective dimensions (less than 10 while the nominal dimension is around 260 with p=0.99) for simple Asian options. The idea of dimension reduction in quasi-Monte Carlo simulation can be actualized using Brownian Bridge. Quasi-random sequences are

used first to determine some points on each path, then the remaining parts of each path are filled using points determined by pseudo random numbers.

We denote $B^{a,b}$ for a Brownian Bridge from a to b over its time interval $[t_i, t_j]$, which represents a Brownian motion B on $[t_i, t_j]$ with $B_{t_i} = a$ and $B_{t_j} = b$. Brownian Bridge Construction (BBC) and Brownian Bridge Discretization (BBD) are two methods of generating Brownian paths (Glasserman, 2003).

Given the initial and terminal value, BBC is an algorithm that generates a discrete Brownian path $B^{a,b} = (B_{t_0}, \dots, B_{t_n})$ recursively. The procedure is as follows:

1. Set $B_{t_0} = a$ and $B_{t_n} = b$.
2. Generate standard normal random numbers $Z_i \sim N(0,1)$ for $i = 1, \dots, n-1$.
3. Calculate $B_{t_{i+1}}$ conditional on B_{t_i} and B_{t_n}, for $i = 0, \dots, n-2$, recursively using the Brownian Bridge formula:

$$B_{t_{i+1}} = \frac{t_n - t_{i+1}}{t_n - t_i} B_{t_i} + \frac{t_{i+1} - t_i}{t_n - t_i} B_{t_n} + \sqrt{\frac{(t_{i+1} - t_i)(t_n - t_{i+1})}{t_n - t_i}} Z_{i+1}. \quad (5)$$

BBD is another method that is based on Brownian Bridge formula. The difference between BBD and BBC is the order in which the points are filled. In BBC, it is clear that points are filled in one behind another to form the path. In BBD, middle points of existing points are filled at each time. This method generates a discrete Brownian path $B^{a,b} = (B_{t_0}, \dots, B_{t_n})$ with n equals a power of 2, and the idea is always to fill in the middle points first. The order of assigning the generated points to the path could be

`

$$\frac{1}{2}n, \frac{1}{4}n, \frac{3}{4}n, \frac{1}{8}n, \frac{3}{8}n, \frac{5}{8}n, \frac{7}{8}n, \frac{1}{16}n, \frac{3}{16}n, \frac{5}{16}n, \frac{7}{16}n, \frac{9}{16}n, \ldots$$

The procedure is as follows:

1. Set $B_{t_0} = a$ and $B_{t_n} = b$.
2. Generate standard normal random numbers $Z_i \sim N(0,1)$ for $i = 1, \ldots, n-1$.
3. Calculate $B_{t_{n\times\frac{1}{2}}}$ conditionally on B_{t_0} and B_{t_n} using Equation (5).
4. Calculate $B_{t_{\frac{n}{4}}}$ conditionally on B_{t_0} and $B_{t_{\frac{n}{2}}}$ and calculate $B_{t_{\frac{3n}{4}}}$ conditionally on $B_{t_{\frac{n}{2}}}$ and B_{t_n}, etc. until all n-1 spot values are calculated.

Comparing with BBC, the first few spot values generated by BBD gives much more sense of what the path looks like. Due to the poor uniformity of low discrepancy sequences in high dimensions, we want to use only d quasi-random numbers to simulate a discrete Brownian path of length L. The d quasi-random numbers would be more effectively capture the important dimensions while using BBD.

For a consistent use in algorithms and implementation, a list of notations is provided below:

L represents the number of time steps for each path;

d represents the dimension of a low-discrepancy sequence;

LD represents the length of the low-discrepancy sequences;

MC represents the number of trajectories of each conditioned process;

N equals $LD \times MC$, represents the total number of paths being simulated;

`

T represents the time at maturity.

Note that d is chosen to be a power of 2 and L is chosen to be a multiple of d, so that each conditioned process has the same length.

To generate Brownian paths, low discrepancy points are used to capture the important dimensions. Random points are then used to fill in the gaps. A detailed algorithm for generating $\mathrm{N} = \mathrm{LD} \times \mathrm{MC}$ Brownian paths using quasi-Monte Carlo simulation is given below. In the remaining part of the thesis, BB-QMC refers to this algorithm.

BB-QMC Algorithm

Algorithm Input: $\mathrm{d}, \mathrm{L}, \mathrm{LD}, \mathrm{MC}$ and T.

Algorithm Output: A matrix, Path, with dimension $(LD \times MC, L + 1)$.

Algorithm:

1. Generate a d-dimensional low-discrepancy sequence of length LD.
2. Apply the BBD method to calculate LD sets of d spot values using the low-discrepancy sequence.

   For $i = 1, \dots, LD$

   a) Generate standard normal random numbers $Z(\, i, j)$ for $j = 1, \dots, d$ using the d-dimensional low-discrepancy sequence from Step 1.

b) Set $B(i,1) = 0$ and $B(i,1+d) = \sqrt{T}Z(i,d)$.

c) Calculate $B(i,\frac{d}{2}+1)$ conditionally on $B(i,1)$ and $B(i,1+d)$ using $Z(i,\frac{d}{2})$ and Equation (5).

d) Calculate $B(i,\frac{d}{4}+1)$ conditionally on $B(i,1)$ and $B\left(i,\frac{d}{2}+1\right)$ using $Z(i,\frac{d}{4})$; and calculate $B(i,\frac{3d}{4}+1)$ conditionally on $B\left(i,\frac{d}{2}+1\right)$ and $B(i,1+d)$ using $Z(i,\frac{3d}{4})$, etc. until all d spot values are created.

3. Apply BBC method to generate MC Brownian Bridges of length $\frac{L}{d}$ conditionally on each set of d spot values.

For $i = 0, \dots, LD-1$

For $k = 0, \dots, d-1$

For $j = 1, \dots, MC$

a) Set $Path\left(i \times MC + j, k \times \frac{L}{d}+1\right) = B(i+1,k+1)$ and $Path\left(i \times MC + j, (k+1) \times \frac{L}{d}+1\right) = B(i+1,k+2)$.

b) Generate $(\frac{L}{d}-1)$ standard normal random variables $Z(l)$ for $l = 1, \dots, \frac{L}{d}-1$.

c) Apply Equation (5) to calculate $Path\left(i \times MC + j, k \times \frac{L}{d} + 1 + l\right)$ conditionally on $Path\left(i \times MC + j, k \times \frac{L}{d} + l\right)$ and $Path\left(i \times MC + j, (k+1) \times \frac{L}{d} + 1\right)$ for $l = 1, \ldots, \frac{L}{d} - 1$.

`

Chapter 3

BBI Quasi-Monte Carlo Method

A high level literature review of variance reduction techniques for quasi-Monte Carlo simulation is provided in Kolkiewicz (2014). A new method of generating Brownian motion sample paths was developed for the purpose of increasing efficiency of simulation methods. The key idea of the new method rests on the intelligent use of the d-dimensional low-discrepancy sequence. This method is referred to as BBI in this thesis. The BBI method of generating Brownian motion paths provides an efficient simulation method to evaluate expectations of the form

$$E\left[G\left(\int_0^T g_1\big(t,W(t)\big)dt\,,\ldots,\int_0^T g_r\big(t,W(t)\big)dt\,,W(\cdot)\right)\right]$$

where $g_1,\ldots,g_r$ are given functions and $W(\cdot)$ is a path of a standard Brownian motion. In the case of $r=1$, Kolkiewicz (2014) proved that for a smooth integrand with certain assumptions, variance of the integrand using the BBI method is of smaller order than the variance of the BB method.

In one Brownian path, consider two consecutive Brownian Bridges, $B^{a,b}$ from time step t_a to t_b and $B^{b,c}$ from time step t_b to t_c constructed as in Step 3(c) of QMC in Appendix A. BBD method is applied using low discrepancy points. More precisely, to generate one Brownian path in a specific simulation problem, one point in the first dimension of a low-discrepancy sequence is used to obtain the value

$B_{t_c}^{b,c} = c$, and one point in the second dimension of the low-discrepancy sequence is used to calculate $B_{t_b}^{a,b} = b$ conditionally on $B_{t_c}^{b,c} = c$. Several paths of the two Brownian bridges are then constructed conditionally on these two terminal values and initial value $B_{t_a}^{a,b} = a$. Putting the two Brownian Bridges, $B^{a,b}$ and $B^{b,c}$ together, $B^{a,c}$ is a Browian Path from time steps t_a to t_c with $B_{t_a}^{a,c} = a$ and $B_{t_c}^{a,c} = c$.

In the BBI method, after $B_{t_c}^{a,c} = c$ is calculated, it is suggested that we construct paths from time steps t_a to t_c conditionally on the integral of the path. Therefore, two issues have to be addressed. The first problem is how to determine the integral of a path so that one can construct that path conditionally on the determined integral? The second issue is how to generate a path while given a pre-determined integral of the path? These two problems are solved in the following two sections. The algorithm of the BBI simulation method is given in Appendix B.

3.1 Integral of Brownian Bridges

Let W_t denote a standard Brownian motion process for $t \in [0,T]$ with $W_0 = 0$, which, in practice, can be simulated over any finite set of times in practice. For $0 < t_1 < t_2 < \cdots < t_n$, the increments $W_{t_1} - W_0, W_{t_2} - W_{t_1}, \dots, W_{t_n} - W_{t_{n-1}}$ are independent and normally distributed with expectation 0. Thus, the expectation $E[W_t] = 0$, and the covariance, $Cov(W_s, W_t)$, can be calculated as follows by letting $0 \le s \le t$

$$Cov(W_s, W_t) = E[W_s W_t] = s.$$

`

Note that, $E[W_s(W_t - W_s)] = 0$ since W_s and $(W_t - W_s)$ are independent normal random variables with mean 0. Hence, we conclude that for $s, t \in [0, T]$,

$$Cov(W_s, W_t) = s \wedge t$$

where $s \wedge t$ denotes the smaller of s and t.

A standard Brownian Bridge from 0 to 0 on time interval $[0, T]$ can be represented as

$$B_t^{0,0} = W_t - \frac{t}{T} W_T, t \in [0, T].$$

The expectation and covariance can be calculated as follows:

$$\begin{aligned} E\left[B_t^{0,0}\right] &= E\left[W_t - \frac{t}{T} W_T\right] \\ &= E[W_t] - \frac{t}{T} E[W_T] \\ &= 0 \end{aligned}$$

for $t \in [0, T]$, and for $s, t \in [0, T]$

$$\begin{aligned} & Cov\left(B_s^{0,0}, B_t^{0,0}\right) \\ &= Cov(W_s - \frac{s}{T} W_T, W_t - \frac{t}{T} W_T) \\ &= Cov(W_s, W_t) - \frac{t}{T} Cov(W_s, W_T) - \frac{s}{T} Cov(W_t, W_T) + \frac{st}{T^2} Cvo(W_T, W_T) \\ &= (s \wedge t) - \frac{t}{T}(s \wedge T) - \frac{s}{T}(t \wedge T) + \frac{st}{T^2}(T \wedge T) \\ &= s \wedge t - \frac{st}{T}. \end{aligned}$$

A Brownian Bridge from a to b on time interval [0,T] can be represented as

`

$$B_t^{a,b} = a + (b-a)\frac{t}{T} + B_t^{0,0}, t \in [0,T]$$

where $B_t^{0,0}$ is a standard Brownian Bridge. For $s, t \in [0,T]$, the expectation and covariance of $B_t^{a,b}$ are computed as follows:

$$\begin{aligned} E\left[B_t^{a,b}\right] &= E\left[a + (b-a)\frac{t}{T} + B_t^{0,0}\right] \\ &= a + (b-a)\frac{t}{T} - E\left[B_t^{0,0}\right] \\ &= a + (b-a)\frac{t}{T} \end{aligned}$$

and

$$\begin{aligned} & Cov\left(B_s^{a,b}, B_t^{a,b}\right) \\ &= Cov(B_s^{a,b} - E\left[B_s^{a,b}\right], B_t^{a,b} - E\left[B_t^{a,b}\right]) \\ &= Cov\left(B_s^{0,0}, B_t^{0,0}\right) \\ &= s \wedge t - \frac{st}{T}. \end{aligned}$$

Note that for $0 < t_1 < t_2 < \cdots < t_n < T$, $B_{t_1}^{a,b}, B_{t_2}^{a,b}, \ldots, B_{t_n}^{a,b}$ are jointly normal since $W_{t_1}, W_{t_2}, \ldots, W_{t_n}, W_T$ are jointly normal.

The integral $A^{a,b}$ of Brownian Bridge $B_t^{a,b}$ over the time interval $[0,T]$ is defined to be

$$A^{a,b} = \int_0^T B_t^{a,b} dt$$

`

$$= \int_0^T a + (b-a)\frac{t}{T}\,dt + \int_0^T B_t^{0,0}\,dt$$

$$= \frac{(a+b)T}{2} + \int_0^T B_t^{0,0}\,dt.$$

Since $B_{t_1}^{a,b}, B_{t_2}^{a,b}, \dots, B_{t_n}^{a,b}$ are jointly normally distributed for $t_a < t_1 < t_2 < \cdots < t_n < t_b$, it can be shown that $A_{t_a,t_b}^{a,b}$ is normally distributed with mean $E[A^{a,b}]$ and variance $Var[A^{a,b}]$ of the following forms:

$$E[A^{a,b}] = a + \frac{(b-a)T}{2} + E\left[\int_0^T B_t^{0,0}\,dt\right]$$

$$= a + \frac{(b-a)T}{2},$$

and

$$Var[A^{a,b}] = E[A^{a,b} - E[A^{a,b}])^2]$$

$$= E\left[\left(\int_0^T B_t^{0,0}\,dt\right)\left(\int_0^T B_s^{0,0}\,ds\right)\right]$$

$$= \int_0^T \int_0^T E\left[B_s^{0,0} B_t^{0,0}\right] ds\,dt$$

$$= \int_0^T \int_0^T Cov\left(B_s^{0,0}, B_t^{0,0}\right) ds\,dt$$

$$= \int_0^T \int_0^T s \wedge t\; ds\,dt - \int_0^T \int_0^T \frac{st}{T}\; ds\,dt$$

$$= \frac{T^3}{3} - \frac{T^3}{4} = \frac{T^3}{12}.$$

`

Therefore, the integral of a Brownian Bridge, $A^{a,b}$ can be determined, in simulation, using the representation

$$A^{a,b} = \mu^{a,b} + \sigma^T z \tag{6}$$

with

$$\mu^{a,b} = \frac{(a+b)T}{2} \tag{7}$$

and

$$\sigma^T = \sqrt{\frac{T^3}{12}} \tag{8}$$

where

- a represents the initial value of the path,
- b represents the terminal value of the path,
- T represents the time length of the path,
- z represents a standard normal random variable.

3.2 Brownian Path Conditional on Integral

This section starts with an important theorem for Gaussian processes presented in Kolkiewicz (2014). A specific version for a Brownian Bridge is given below followed by a proof.

`

Theorem. *Let $B_t^{a,b}, t \in [0,T]$ be a Brownian Bridge from a to b. Then, the law of the process*

$$Z_t^{a,b} = B_t^{a,b} - \frac{6t(T-t)}{T^3}\left[\int_0^T B_s^{a,b} ds - \alpha^{a,b}\right], t \in [0,T]$$

has the same law as the process $B_t^{a,b}, t \in [0,T]$ given that $\int_0^T B_s^{a,b} ds = \alpha^{a,b}$.

Proof. Looking at the covariance between the process $Z_t^{a,b}$ and $\int_0^T B_s^{a,b} ds$. We have

$$Cov\left(Z_t^{a,b}, \int_0^T B_s^{a,b} ds\right)$$

$$= Cov\left(B_t^{a,b} - \frac{6t(T-t)}{T^3}\int_0^T B_s^{a,b} ds + \frac{6t(T-t)}{T^3}\alpha^{a,b}, \int_0^T B_s^{a,b} ds\right)$$

$$= Cov\left(B_t^{a,b}, \int_0^T B_s^{a,b} ds\right) - \frac{6t(T-t)}{T^3} Cov\left(\int_0^T B_t^{a,b} dt\,, \int_0^T B_s^{a,b} ds\right)$$

$$+ Cov\left(\frac{6t(T-t)}{T^3}\alpha^{a,b}, \int_0^T B_s^{a,b} ds\right)$$

$$= E\left[\left(B_t^{a,b} - E\left[B_t^{a,b}\right]\right)\left(\int_0^T B_s^{a,b} ds - E\left[\int_0^T B_s^{a,b} ds\right]\right)\right]$$

$$- \frac{6t(T-t)}{T^3} E\left[\left(\int_0^T B_t^{a,b} dt - E\left[\int_0^T B_t^{a,b} dt\right]\right)\left(\int_0^T B_s^{a,b} ds - E\left[\int_0^T B_s^{a,b} ds\right]\right)\right]$$

$$+ 0$$

$$= E\left[B_t^{0,0}\left(\int_0^T B_s^{0,0} ds\right)\right] - \frac{6t(T-t)}{T^3} E\left[\left(\int_0^T B_t^{0,0} dt\right)\left(\int_0^T B_s^{0,0} ds\right)\right]$$

$$= \int_0^T E\left[B_s^{0,0} B_t^{0,0}\right] ds - \frac{6t(T-t)}{T^3} \times \frac{T^3}{12}$$

$$= \int_0^T s \wedge t \, ds - \int_0^T \frac{st}{T}\, ds - \frac{t(T-t)}{2}$$

`

$$= \int_0^t s\, ds + \int_t^T t\, ds - \frac{tT}{2} - \frac{t(T-t)}{2}$$

$$= 0.$$

Hence, their independence implies that, for any fixed $\alpha^{a,b}$, the law of the process $Z_t^{a,b}$ has the same law as the process $B_t^{a,b}, t \in [0,T]$ given $\int_0^T B_s^{a,b} ds = \alpha^{a,b}$. □

In simulation, a standard Brownian Bridge, $B_t^{0,0}$, will be constructed to generate the Brownian Bridge, $B_t^{a,b}$, conditionally on $\int_0^T B_s^{a,b} ds = \alpha^{a,b}$ using the law of the process

$$B_t^{a,b} = a + (b-a)\frac{t}{T} + B_t^{0,0} - \frac{6t(T-t)}{T^3}\left[\frac{(a+b)T}{2} + \int_0^T B_s^{0,0} ds - \alpha^{a,b}\right], t \in [0,T] \quad (9)$$

where $\alpha^{a,b}$ represents the conditioning integral of the path.

As introduced at the beginning of this chapter, the BBI quasi-Monte Carlo method uses the same number of conditioning variables as the BB-QMC method. The implementation results by Kolkiewicz (2014) indicate a much better performance, and it is believed that the proposed set of conditioning variables would capture a significantly larger amount of variability than that of the BB-QMC method when the dimension increases.

An algorithm for generating Brownian paths using BBI is given below. In the remaining part of the thesis, BBI-QMC refers to this algorithm. There is a new input parameter for the simulation algorithm, denotes dim, which represents the number

of independent stochastic processes in the simulation. Therefore, the results can be used to simulate price paths under stochastic interest rate and/or stochastic volatility models, prices of multiple correlated assets, etc.

BBI-QMC Algorithm

Algorithm Input:

dim represents the number of independent stochastic processes;

d represents the dimension of the simulation problem for each path;

L represents the number of the total time steps for each path;

LD represents the length of the low-discrepancy sequence;

MC represents the number of trajectories of each conditioned process;

T represents the time at maturity.

Algorithm Output:

A matrix, $Paths$, with dimension $(LD \times MC, L+1, dim)$.

Algorithm:

1. Generate a $(d \times dim)$-dimensional low-discrepancy sequence of length LD.
2. Apply the BBD method to calculate LD sets of $\frac{d}{2}$ spot values using first half of the low-discrepancy sequence.

   For $i = 1, \dots, LD$

   For $z = 1, \dots, dim$

`

a) Generate standard normal random numbers $Z(\ i,j)$ for $j = 1, \dots, (d \times dim)$ using the $(d \times dim)$-dimensional low-discrepancy sequence generated in Step 1.

b) Set $B(\ i,1,z) = 0$ and $B\left(i, 1+\frac{d}{2}, z\right) = \sqrt{T}Z\left(i, \frac{d}{2} + d(z-1)\right)$.

c) Calculate $B\left(\ i, \frac{d}{4}+1, z\right)$ conditionally on $B(\ i,1,z)$ and $B\left(i, 1 + \frac{d}{2}, z\right)$ using $Z\left(i, \frac{d}{4} + d(z-1)\right)$ and Equation (5).

d) Calculate $B\left(\ i, \frac{d}{8}+1, z\right)$ conditionally on $B(i,1)$ and $B\left(i, \frac{d}{4} + 1, z\right)$ using $Z\left(i, \frac{d}{8} + d(z-1)\right)$; and calculate $B\left(\ i, \frac{3d}{8}+1, z\right)$ conditionally on $B\left(\ i, \frac{d}{4}+1, z\right)$ and $B\left(i, 1+\frac{d}{2}, z\right)$ using $Z\left(i, \frac{3d}{8} + +d(z-1)\right)$, etc. until all $\frac{d}{2}$ spot values are created for each path.

3. Compute the conditioning integrals using the second half of the low-discrepancy sequence.

   For $i = 1, \dots, LD$

   For $z = 1, \dots, dim$

   For $j = 1, \dots, \frac{d}{2}$

a) Calculate $\mu(i,j,z)$ and $\sigma(i,j,z)$ using Equation (7) and Equation (8), with initial value $B(i,j,z)$, terminal value $B(i,j+1,z)$ and time length $\frac{2T}{d}$.

b) Calculate $A(i,j,z)$ using Equation (6), $Z\left(i,\frac{d}{2}+j+d(z-1)\right)$ and $\mu(i,j,z)$ and $\sigma(i,j,z)$ determined in Step 3(a).

4. Apply the BBC method to generate MC paths with length $\frac{2L}{d}$ conditionally on each set of $\frac{d}{2}$ spot values and $\frac{d}{2}$ integrals.

   For $i = 0, \dots, LD-1$

   For $k = 0, \dots, \frac{d}{2}-1$

   For $j = 1, \dots, MC$

   For $z = 1, \dots, dim$

   a) Set $Path\left(i \times MC + j, k \times \frac{2L}{d}+1, z\right) = B(i+1,k+1,z)$ and $Path\left(i \times MC + j, (k+1) \times \frac{2L}{d}+1, z\right) = B(i+1,k+2,z)$.

   b) Generate $(\frac{2L}{d}-1)$ standard normal random variables $Z(l)$ for $l = 1, \dots, \frac{2L}{d}-1$.

c) Apply Equation (9) to calculate $Path\left(i \times MC + j, k \times \frac{2L}{d} + 1 + l, z\right)$ using initial value equals $Path\left(i \times MC + j, k \times \frac{2L}{d} + l\right)$, terminal value equals $Path\left(i \times MC + j, (k+1) \times \frac{2L}{d} + 1\right)$, time length equals $\frac{2T}{d}$ and conditioning integral equals $A(i+1, k+1, z)$ for $l = 1, \ldots, \frac{2L}{d} - 1$.

`

Chapter 4

An Application of the BBI Quasi-Monte Carlo Method to Asian Option

Here we discuss a Matlab implementation of the algorithm for pricing Asian options under the Heston model. The algorithm is based on the BBI-QMC method presented in the previous chapter. We discuss the performance of Matlab's quasi-random sequence generators in Section 4.1. Section 4.2 compares Moro's normal inverse algorithm with Matlab's built-in function "norminv()". A simple example of Brownian Bridges generated by BBI is given in Section 4.3. Section 4.4 gives parameter settings of the implementation.

4.1 Low Discrepancy Sequences

The first step in our simulation algorithm includes the generation of a multi-dimensional random standard uniform sequence. The function "*rand()*" in Matlab is a pseudo random number generator, so that the resulting sequence of numbers looks random and uniformly distributed on [0,1]. Figure 1 plots a sample output of a 2-dimensional pseudo-random standard uniform sequence of length 1000.

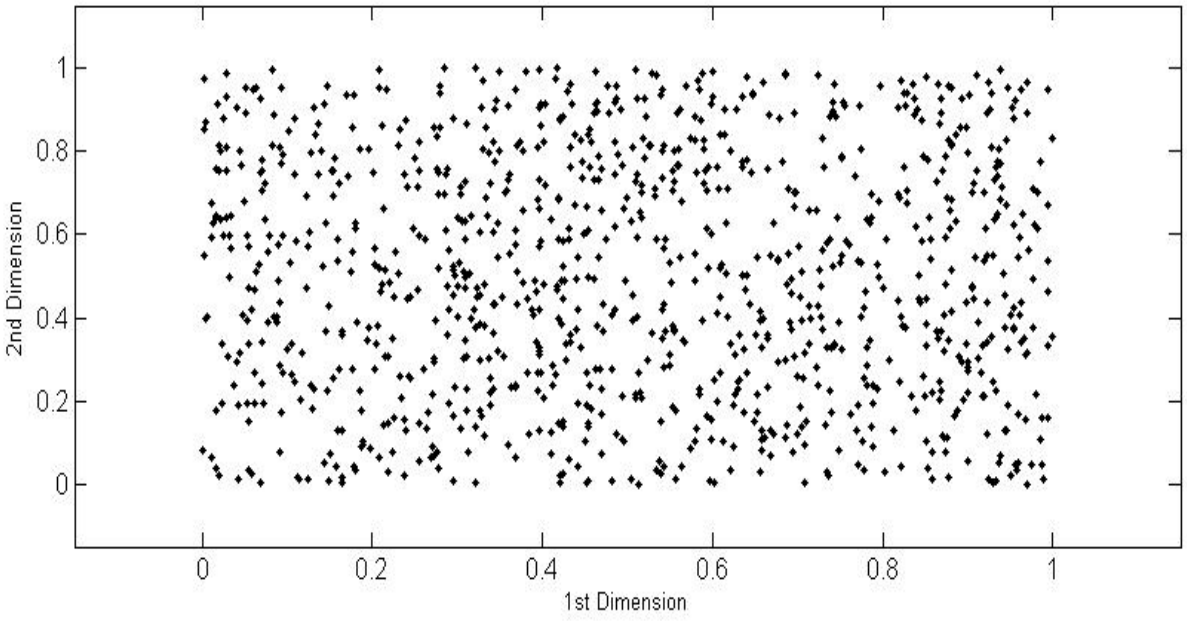

Figure 1: A 2-Dimensional Pseudo-Random Uniform Sequence of Length 1000

There are two quasi-random sequences available in Matlab built-in library[1], Halton sequences (Halton, 1960) and Sobol sequences (Sobol, 1976). The two sequences are introduced and tested in this section before the actual implementation.

The Van der Corput sequence is the first one dimensional low discrepancy sequence. For an integer $p > 0$, integer $n > 0$ can be represent uniquely as

$$n = \sum_{i=0}^{\lfloor \log_p n \rfloor} a_i(n) p^i .$$

The n-th number of the Van der Corput sequence in base p can be represented as

[1] Please refer to the following link for more details of function settings: "*http://www.mathworks.com/help/stats/quasi-random-numbers.html*".

$$VC_p(n) = \sum_{i=0}^{\lfloor \log_p n \rfloor} \frac{a_i(n)}{p^{i+1}}$$

where $\lfloor x \rfloor$ represents the largest integer less than or equal to x.

The Halton sequence generalizes the Van der Corput sequence to higher dimensions. Let the ordered set $\{p_1, p_2, \dots\}$ be the set of all prime numbers in increasing order(e.g. $p_1 = 2, p_3 = 5$). An s-dimensional Halton sequence with length n can be represented as follows:

$$\begin{bmatrix} \left(VC_{p_1}(1), VC_{p_1}(2), \dots, VC_{p_1}(n)\right) \\ \left(VC_{p_2}(1), VC_{p_2}(2), \dots, VC_{p_2}(n)\right) \\ \vdots \\ \left(VC_{p_s}(1), VC_{p_s}(2), \dots, VC_{p_s}(n)\right) \end{bmatrix}.$$

An s-dimensional Sobol sequence is generated from an s-dimensional binary fractions, called direction numbers. The Sobol sequence is not uniquely defined until all of the direction numbers are defined. Antonov and Saleev (1997) proposed an efficient implementation for generation of Sobol sequence. Consider the primitive polynomial in dimension i over the field F_2 with elements {0,1} as

$$P^{(i)}(x) = x^q + p_1 x^{q-1} + \cdots + p_{q-1} x + 1.$$

The direction numbers in dimension i are generated as

$$\left(v^{(i)}(1), v^{(i)}(2), \dots\right)$$

with the recurrence relation

`

$$v^{(i)}(k) = p_1 v^{(i)}(k-1) \oplus p_2 v^{(i)}(k-2) \oplus \ldots \oplus p_{q-1} v^{(i)}(k-q+1) \oplus v^{(i)}(k-q)$$

$$\oplus \left(\frac{v^{(i)}(k-q)}{2^q} \right), \qquad i > q$$

where $\oplus$ denotes the exclusive-or operation. The initial numbers are

$$v_j = \frac{b_j}{2^j}, \qquad j = 1, \ldots, q$$

with random odd integers $0 < b_j < 2^j$. Recall the unique representation of integer $n > 0$ as

$$n = \sum_{i=0}^{\lfloor \log_2 n \rfloor} a_i(n) 2^i .$$

The n-th number of the Sobol sequence $x^{(i)}(n)$ in dimension i is generated as

$$x^{(i)}(n) = a_1 v^{(i)}(1) \oplus a_2 v^{(i)}(2) \oplus \ldots \oplus a_{\log_2 n} v^{(i)}(\log_2 n).$$

The s-dimensional Sobol sequence with length n can be represented as follows:

$$\begin{bmatrix} \left(x^{(1)}(1), x^{(1)}(2), \ldots, x^{(1)}(n) \right) \\ \left(x^{(2)}(1), x^{(2)}(2), \ldots, x^{(2)}(n) \right) \\ \vdots \\ \left(x^{(s)}(1), x^{(s)}(2), \ldots, x^{(s)}(n) \right) \end{bmatrix}.$$

Figure 2 gives the plot of a 2-dimensional Halton sequence of length 1000, and Figure 3 shows the plot of a Sobol sequence. The points on both Figure 2 and Figure 3 look more evenly distributed than the points on Figure 1, which confirm theoretical properties of Sobol and Halton sequences.

`

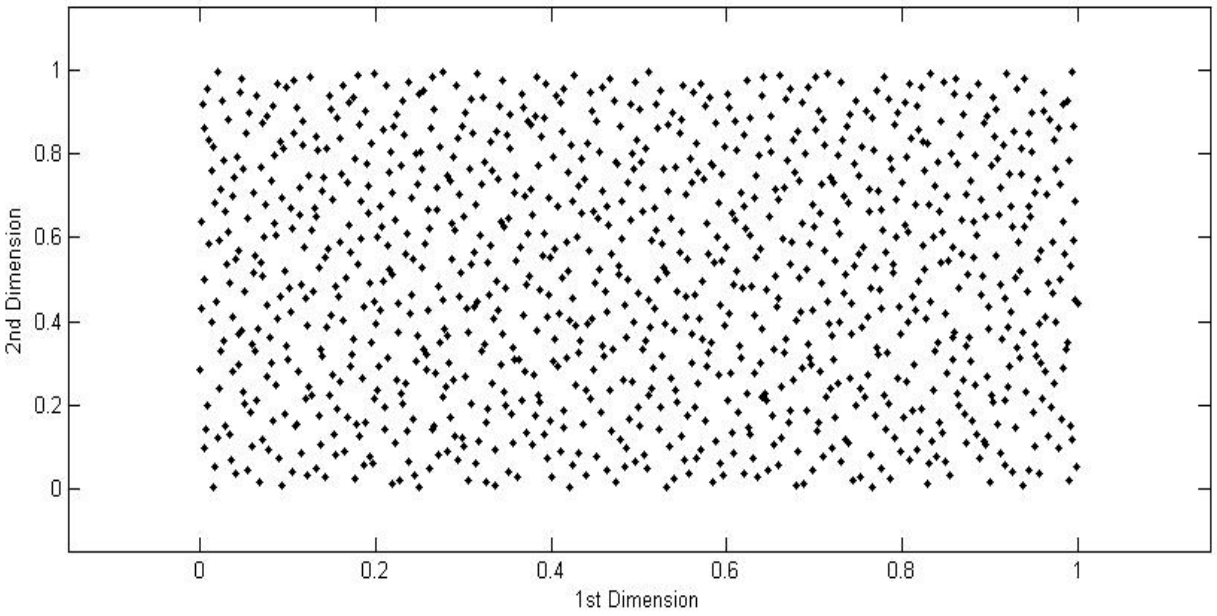

Figure 2: A 2-Dimensional Halton Sequence of Length 1000

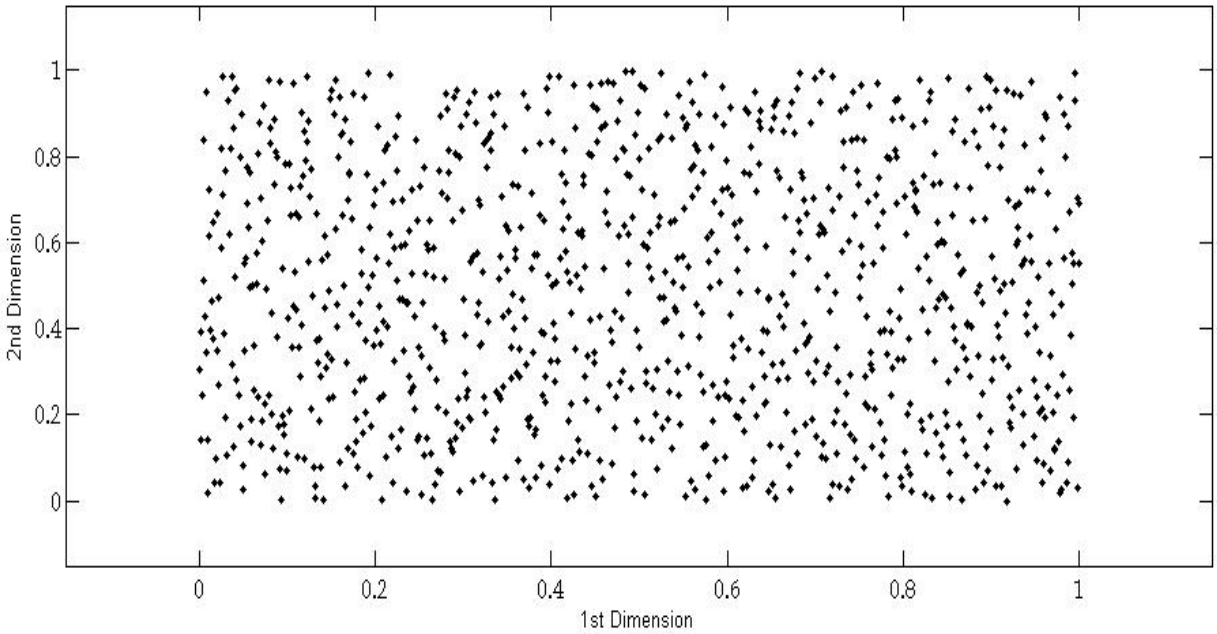

Figure 3: A 2-Dimensional Sobol Sequence of Length 1000

`

Each dimension of a Halton sequence is generated using a different prime number p. Press (1992) argued that every time the length n of Halton sequence increases by one, n's reversed fraction becomes a factor of p finer-meshed. As a result, the generation process fills in all of the points on a sequence of finer and finer Cartesian grids. On the other hand, each dimension of Sobol sequence is generated using a different primitive polynomial. Press (1992) list 150 primitive polynomials which allow the construction of 150-dimensional Sobol sequences. A simulation for an underlying asset price under stochastic volatility will need a quasi-random sequence of dimension 16 using BBI-QMC method, considering an 8-dimensional sequence for each stochastic process. As introduced in Section 2.4, Galanti and Jung (1997) claimed that Sobol sequences outperform Halton sequences in high dimensions (greater than 20).

For comparison of Halton and Sobol sequences in Matlab implementation, Figure 4 and Figure 5 give an example of 16-dimensional Halton and Sobol sequences of length 1000. Each graph contains 256 plots. The 16 bar charts on the diagonal show the distribution of numbers in each dimension. There are 120 small graphs on both the left side and the right side of the diagonal. Each small graph plots one pair from the 16 dimensions like what have done in Figure 1 to 3. For a better performance, both the two 16-dimensional quasi-random sequences are

`

already scrambled using the Matlab "*scramble()*" function. For more details, please refer to ' *http://www.mathworks.com/help/stats/qrandset.scramble.html* '.

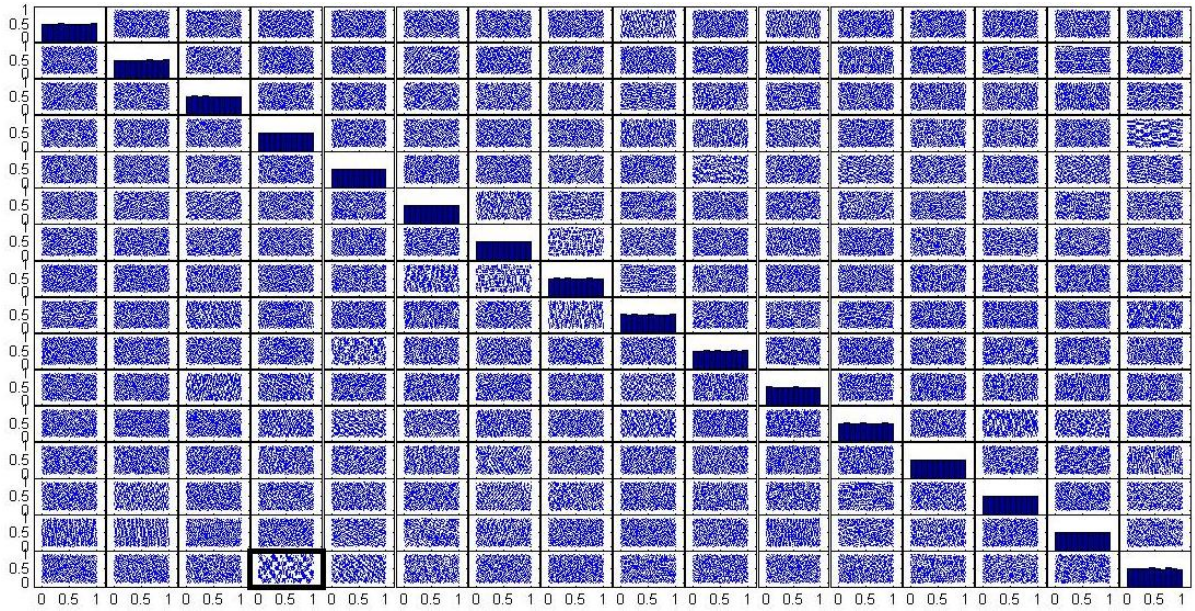

Figure 4: Overview of a 16-Dimensional Holton Sequence of length 1000

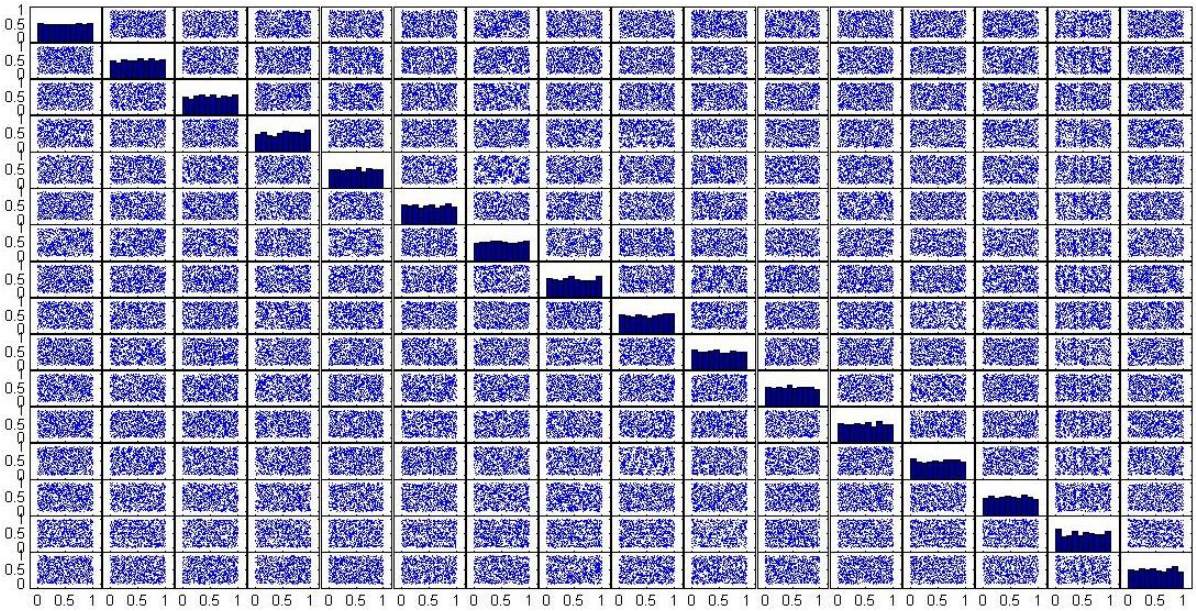

Figure 5: Overview of a 16-Dimensional Sobol Sequence of length 1000

`

The graphs suggest that Sobol sequence performs better when the dimension is a bit higher, since some unsatisfactory small graphs can be found in Figure 4 including the highlighted graph at row 16 column 4. Figure 6 shows a clear version of the highlighted graph. The distribution bar charts of the Holton sequence at dimension 4 and dimension 16 are given as well in Figure 7 and Figure 8. From Figure 7 and Figure 8, we can see that the values at both the 4th and 16th dimensions are evenly distributed. Therefore, the gaps in Figure 6 indicate that the values in 4th and 16th dimensions are less independent.

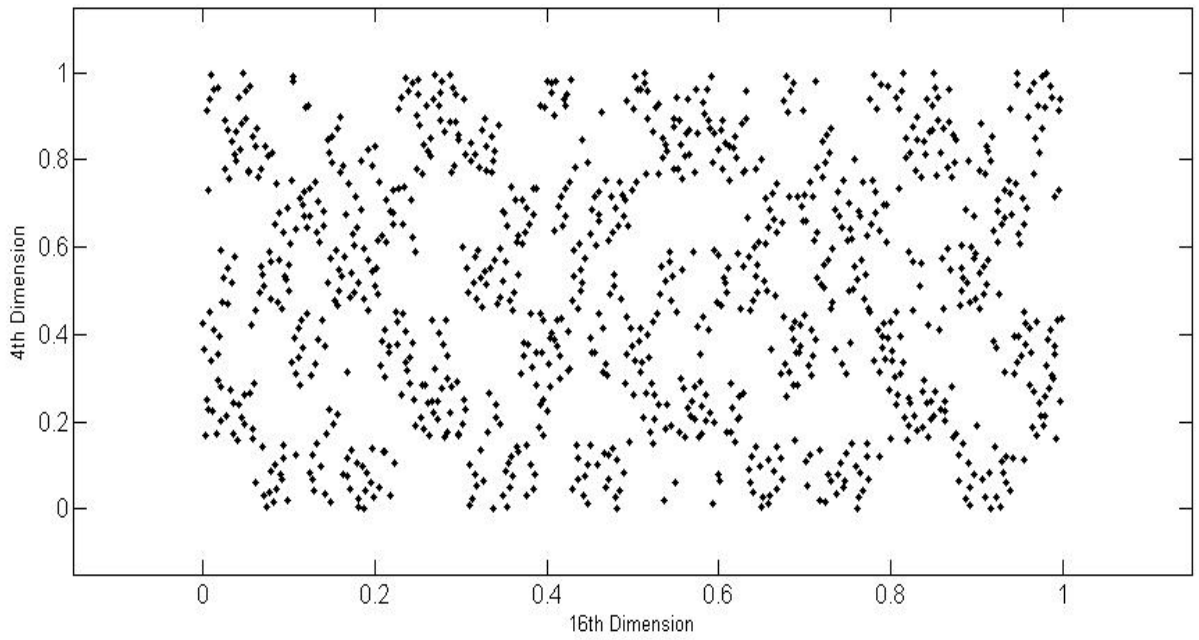

Figure 6: Dimensions 4×16 of the sample Holton sequence

`

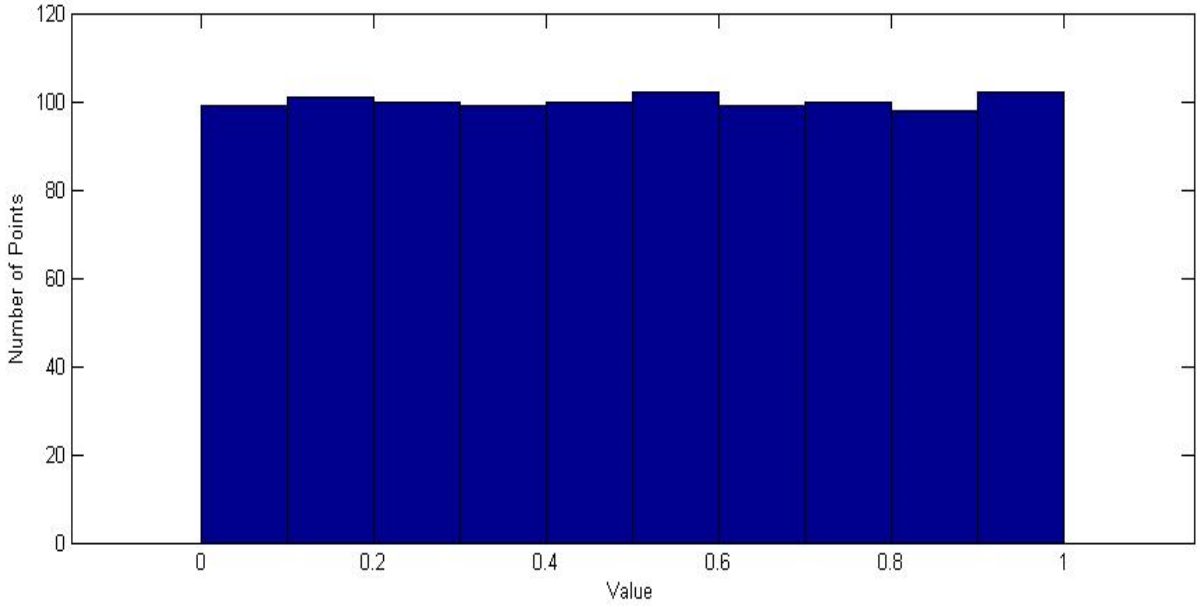

Figure 7: Value Distribution of the Sample Holton sequence, Dimension 4

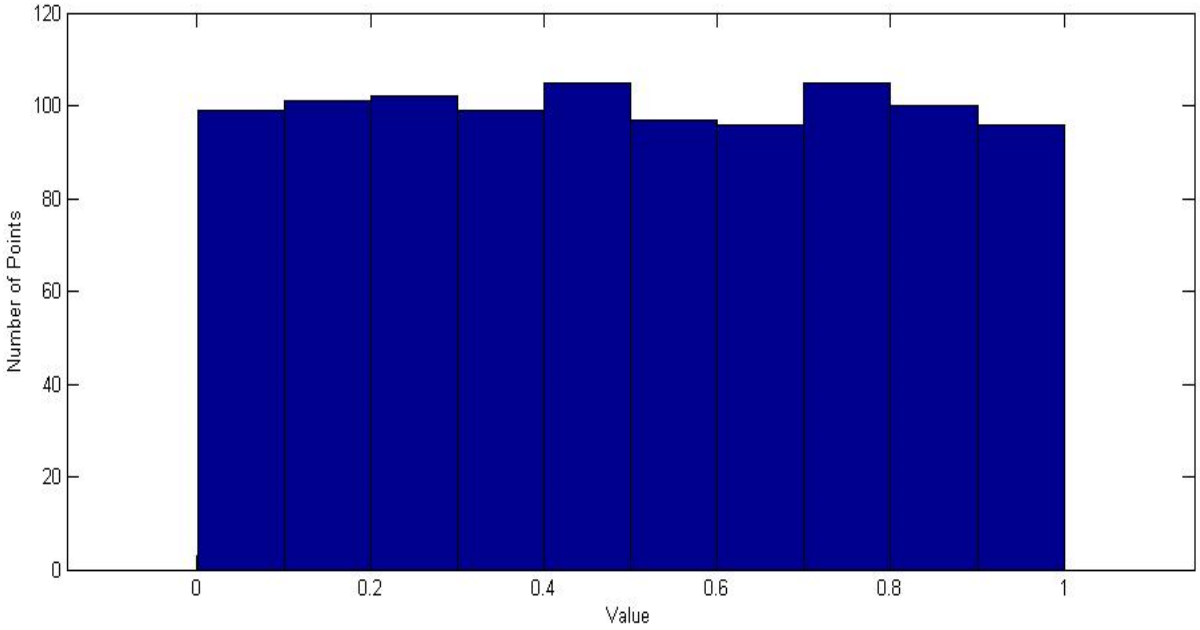

Figure 8: Value Distribution of the Sample Holton sequence, Dimension 16

`

For implementation efficiency, Holton sequence is used for dimension less than 10, and Sobol sequence is selected for dimension higher than 10.

4.2 Probit Function

The cumulative distribution function of the standard normal distribution is:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

In the simulation, we let $\Phi(x)$ to be uniformly distributed on [0,1]. Therefore, for each value u from a low discrepancy sequence, there is a corresponding $x = \Phi^{-1}(u)$ sample from the standard normal distribution, where the probit function Φ^{-1} is the inverse of the cumulative distribution function Φ. Since the probit function is not available in closed form, numerical algorithms are widely available in software such as Matlab.

The best known method of numerically implementing the probit function is by using the Box Muller algorithm. Galanti and Jung (1997) suggested that the famous Box Muller algorithm (Press at al. 1992) damages the low discrepancy properties of the sequences, and the traditional Beasley-Springer (1977) algorithm has poor performance for the tails of the normal distribution. We implemented Moro's (1995) algorithm which uses Beasley-Springer algorithm for the central part of the normal distribution, and a truncated Chebyshev series to model the tails of the distribution. Moro's algorithm for implementation are given as follows.

`

Moro's Normal Inverse Algorithm

Algorithm Input: $x \in [0,1]$.

Algorithm Output: $\Phi^{-1}(x) \in (-\infty, +\infty)$.

Algorithm:

1. The central part of the distribution is when $x \in [0.08,0.92]$. Beasley and Springer algorithm is applied to this region using formula:

$$\Phi^{-1}(x) = \frac{\sum_{n=0}^{3} a_n \left|x - \frac{1}{2}\right|^{2n+1}}{1 + \sum_{n=0}^{3} b_n \left|x - \frac{1}{2}\right|^{2n}}.$$

2. The tail part of the distribution is when $x \in [0,0.08) \cup (0.92,1]$. A truncated Chebyshev series is used to model this region using formula:

$$\Phi^{-1}(x) = \begin{cases} \sum_{n=0}^{8} c_n T_n(x) & x > 0.92 \\ -\sum_{n=0}^{8} c_n T_n(1-x) & x < 0.08 \end{cases}$$

where

$$T_n(x) = [\ln(-\ln(1-x))]^n.$$

The parameters a_n, b_n and c_n are given below.

n	a_n	b_n	c_n
0	2.50662823884	-8.47351093090	0.3374754822726147
1	-18.61500062529	23.08336743743	0.9761690190917186
2	41.39119773534	-21.06224101826	0.1607979714918209

`

3	-25.44106049637	3.13082909833	0.0276438810333863
4	N/A	N/A	0.0038405729373609
5	N/A	N/A	0.0003951896511919
6	N/A	N/A	0.0000321767881768
7	N/A	N/A	0.0000002888167364
8	N/A	N/A	0.0000003960315187

The Matlab code for Moro's algorithm is provided in Appendix C. Moro (1995) claims a maximum absolute error of 3×10^{-9} over the range $x \in [\Phi(-7), \Phi(7)]$.

We can now compare the standard normal quasi-random sequence and the standard normal random sequence. Scrambled Halton sequence of length 1000, Pseudo-random sequence of length 1000 and 10000 are used with Moro's algorithm for comparison. Statistical properties are presented in Table 1 below.

Table 1 Comparison of Standard Normal Quasi-Random and Pseudo-Random Sequences

	Standard Normal	Quasi-random Sequence of Length 1000	Pseudo-random Sequence of Length 1000	Pseudo-random Sequence of Length 10000
Mean	0	-0.0000184	0.04360068	0.02226267
Standard Deviation	1	0.99878002	0.98199416	1.01528914
Variance	1	0.99756152	0.96431253	1.03081205
Skewness	0	-0.0006303	-0.04438527	0.001724717
Kurtosis	3	2.9634615	2.8296301	2.94454714

It can be seen from Table 1 that the quasi-random sequence is much more efficient in practical implementation of normality. Two dimensional sequences are plotted for a visual demonstration.

`

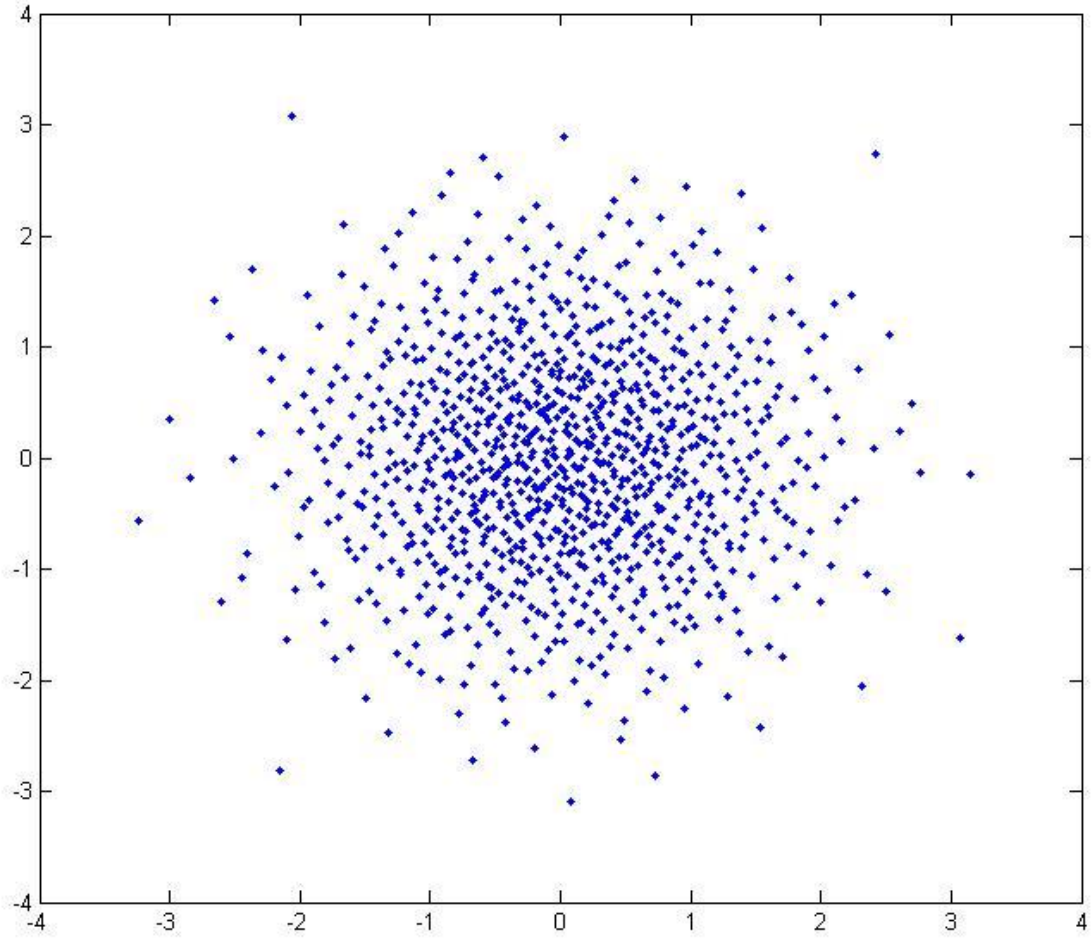

Figure 9: 2-Dimensional Standard Normal Quasi-Random Numbers of length 1000

`

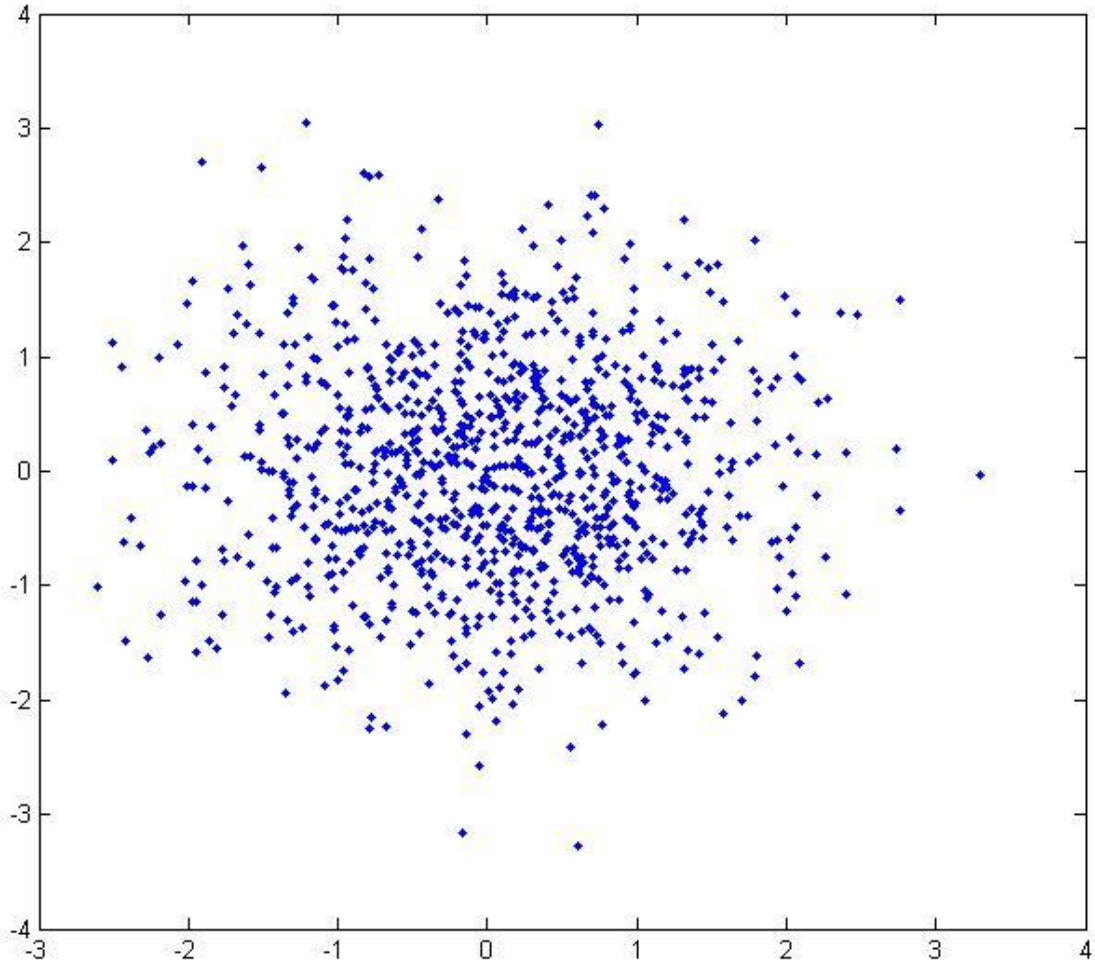

Figure 10: 2-Dimensional Standard Normal Random Numbers of length 1000

`

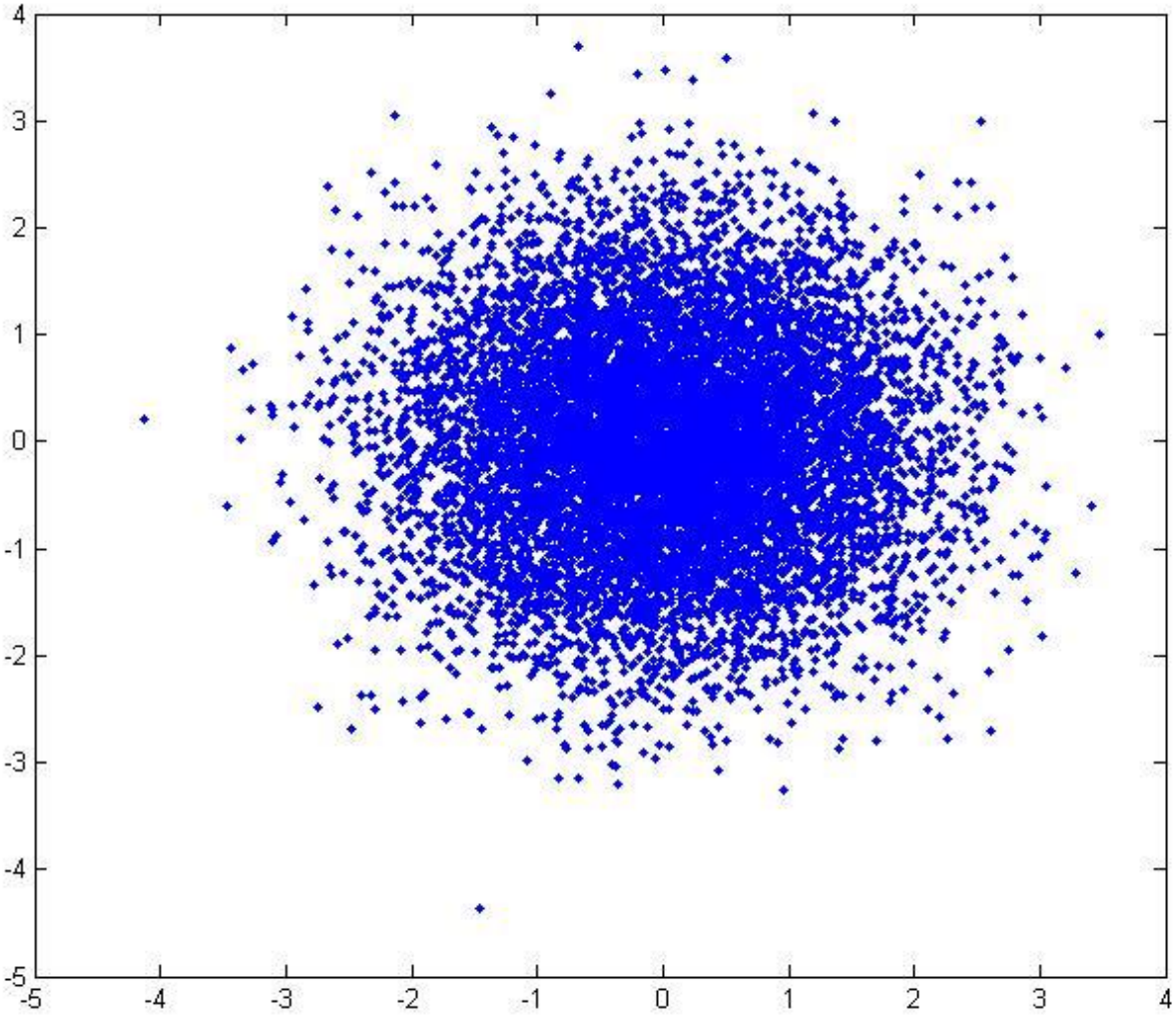

Figure 11: 2-Dimensional Standard Normal Random Numbers of length 10000

Comparison between Figure 9 and Figure 10 strongly suggests a much lower discrepancy of quasi-random sequences. Results in Figure 9 and Figure 11 are comparable. The graphs confirm the nice performance of the quasi-random sequence.

`

4.3 Generation of Brownian Paths for BB and BBI methods

QMC is introduced before BBI-QMC for a better understanding of the latter. Both algorithms are given in Section 2.5 and Section 3.3. Input parameters of BBI-QMC simulation are introduced again with selection tips:

1. d represents the dimension of the low-discrepancy sequence for each path. That means each Brownian path is constructed conditionally on d variables. Usually, $d = 2$ or $d = 4$ is good enough for a fast convergence. Wang and Tan (2012) concluded that BBD methods results in the effective dimension of 6 while the nominal dimension is 260 with p=0.99 for simple Asian options. In this thesis d is chosen to be 32.
2. T represents the time at maturity. For example, $T = 1$ means the maturity is 1 year from now. In this thesis T is chosen to be 1.
3. L represents the number of the total time steps for each path. For example, one can choose $L = 260 \times T$ to represent the total number of business days during the contract. We assumed that L is always chosen to be a multiple of d, so that the Brownian Bridges constructed conditionally on the d variables could have the same length. Thus, in this thesis L is chosen to be 320, so that each Brownian path is of length 320 generated using 32 low discrepancy points.

`

4. LD represents the length of the low-discrepancy sequence, and MC represents the number of trajectories of each conditioned process. Thus, LD sets of conditional variables are first determined and MC paths are generated conditionally on each set. Therefore, a total number of $N = LD \times MC$ Brownian paths are constructed to simulate one Brownian motion.

For demonstration, Figure 12 shows 10 simulated Brownian paths conditioning on one middle value and one terminal value using QMC method (i.e. $d = 2$, $LD = 1$ and $MC = 10$). For comparison, Figure 13 gives 10 simulated paths conditioning on one terminal value and one pre-determined integral using BBI-QMC (i.e. $d = 2$, $LD = 1$ and $MC = 10$). The number of time steps is chosen to be 1000 with maturity $T = 1$, so that the paths in Figure 12 and Figure 13 look smooth. The two graphs together give an intuitive grasp of the BBI-QMC algorithm. Both the two path generation methods are coded in Matlab and attached in Appendix B and Appendix C.

`

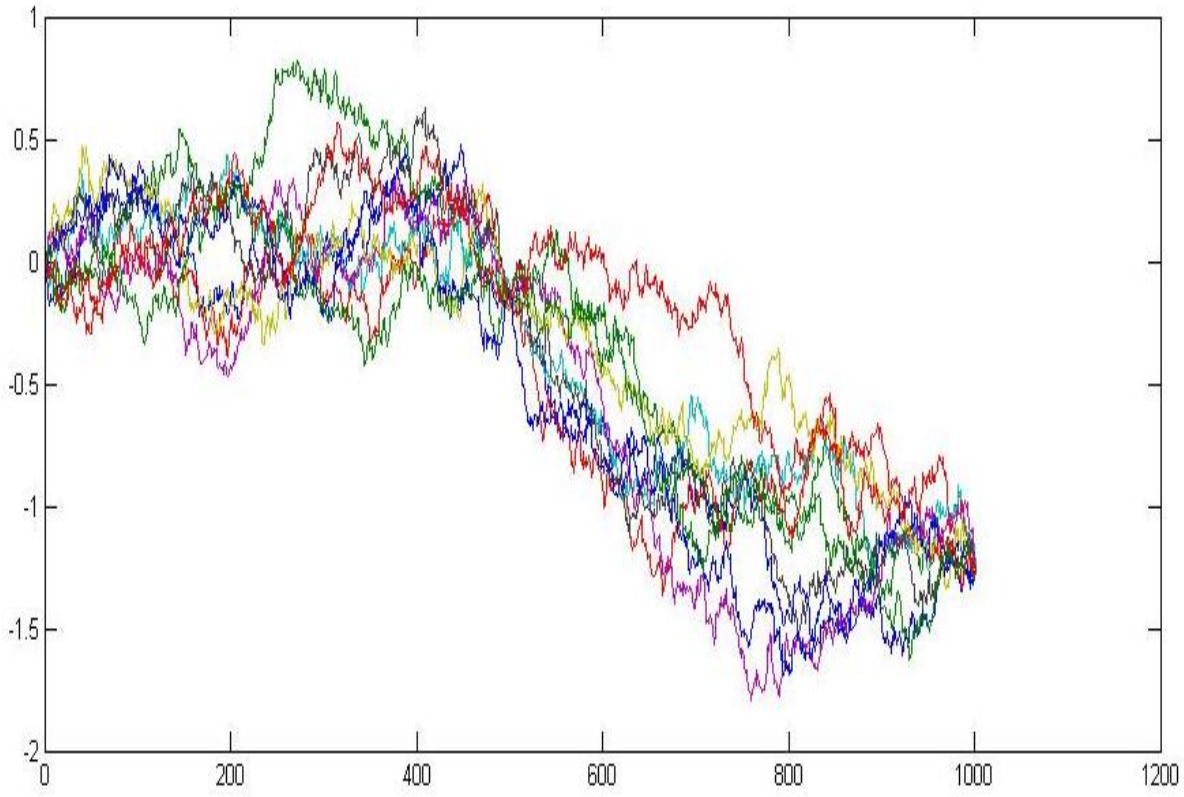

Figure 12: 10 Simulated Paths using BB-QMC method, d=2

`

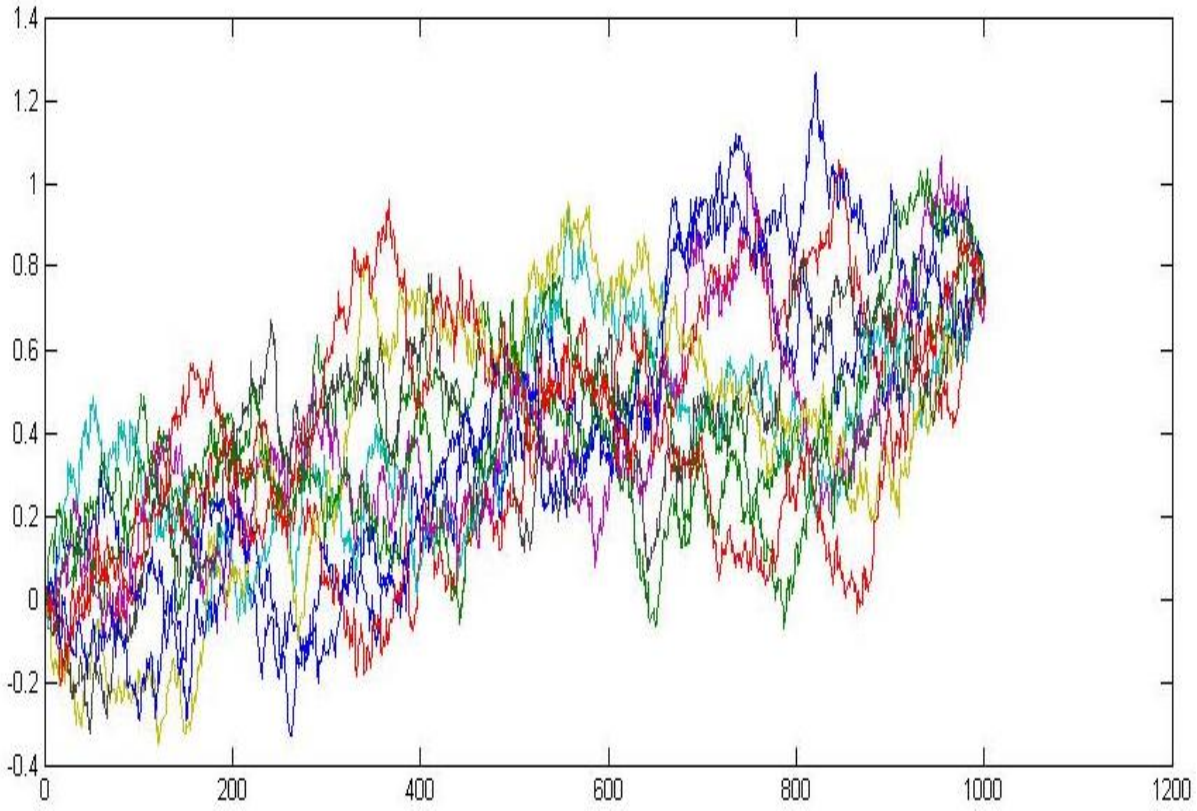

Figure 13: 10 Simulated Paths using BBI-QMC, d=2

4.4 Pricing Asian Option under Heston's Model

We have implemented the BBI method to price a European arithmetic Asian call option with payoff function as follows:

$$\text{Payoff} = \max[(A_T - K)\,, 0].$$

Wang and Tan (2012) illustrate the impact of path generation methods using three choices of averaging methods. For $A_T = \sum_{i=1}^{L} w_i^{(\cdot)} S_{t_i}$, the three weights are defined as follows:

$$w_i^A = \frac{1}{L},$$

`

$$w_i^B = b2^{-i} \text{ and}$$

$$w_i^C = \frac{c(-1)^{i-1}}{i}$$

for $i = 1, \dots L$. Normalization constants b and c are chosen so that $\sum_{i=1}^{L} w_i^B = 1$ and $\sum_{i=1}^{L} w_i^B = 1$. Note that the second and third averaging methods have more weights on the beginning of the underlying price process. Their results indicate that the performance of BBD is even worse than standard Brownian path construction while $\{w_i^B\}$ and $\{w_i^C\}$ are implemented. The implementation in this thesis considers even weight $\{w_i^A\}$ only, so that

$$A_T = \frac{1}{L}\sum_{i=0}^{L} S_i.$$

Consider the continuous form of Equation (3) and Equation (4),

$$S_t = S_0 \exp\left(\left(r - \frac{1}{2t}\int_0^t V_s ds\right)t + \rho\int_0^t \sqrt{V_s}dW_s + \sqrt{1-\rho^2}\int_0^t \sqrt{V_s}dZ_s\right)$$

$$= S_0 \exp\left(\left(r - \frac{\rho}{\delta}\alpha\overline{\sigma}^2\right)t + \frac{\rho}{\delta}(V_t - V_0) + \left(\frac{\rho}{\delta}\alpha - \frac{1}{2}\right)\int_0^t V_s ds \right.$$

$$\left. + \sqrt{1-\rho^2}\int_0^t \sqrt{V_s}dZ_s\right) \qquad (10)$$

The Asian call option payoff can be represented as the expectation of a function G of $\{V_t\}$ and $\left\{\int_0^t V_s ds\right\}$ below

$$\text{Payoff} = E[(A_T - K)^+]$$

`

$$= E\left[E\left[\left(\frac{1}{T}\int_0^T S_t dt - K\right)^+ \mid \{V_t\}\right]\right]$$

$$= E\left[E\left[\left(\frac{S_0}{T}\int_0^T \exp\left\{\left(r - \frac{\rho}{\delta}\alpha\overline{\sigma}^2\right)t + \frac{\rho}{\delta}(V_t - V_0) + \left(\frac{\rho}{\delta}\alpha - \frac{1}{2}\right)\int_0^t V_s ds\right.\right.\right.$$

$$\left.\left.\left. + \sqrt{1-\rho^2}\int_0^t \sqrt{V_s} dZ_s\right\} dt - K\right)^+ \mid \{V_t\}\right]\right]$$

$$= E\left[G\left(\{V_t\}, \left\{\int_0^t V_s ds\right\}\right)\right].$$

In the Heston model, the process $\{V_t\}$ follows the diffusion process

$$dV_t = \alpha\left(\overline{\sigma}^2 - V_t\right)dt + \delta\sqrt{V_t}dW_t, \qquad t \geq 0.$$

Let the transformation be $\beta(x) = \frac{2}{\delta}\sqrt{x}$ so that by applying the Ito's formula we have

$$d\beta(V_t) = \frac{1}{\delta\sqrt{V_t}}dV_t - \frac{\delta}{4\sqrt{V_t}}dt$$

$$= \mu(V_t)dt + dW_t, \qquad \text{and}$$

$$\mu(x) = \frac{\alpha\left(\overline{\sigma}^2 - x\right)}{\delta\sqrt{x}} - \frac{\delta}{4\sqrt{x}}. \tag{12}$$

From the Girsanov theorem, the likelihood ratio of the measure generated by the process $\{\beta(V_t)\}$ with respect to the measure induced by $\left\{\frac{2}{\delta}\sqrt{V_0} + W(t)\right\}$ is

$$L(\omega) = \exp\left[\int_0^T \mu\left(\omega(u)\right)d\omega(u) - \frac{1}{2}\int_0^T \mu^2\left(\omega(u)\right)du\right].$$

`

Let

$$D(x) = \int^{x} \mu(z)dz$$

$$= 2\left(\frac{\alpha\overline{\sigma}^2}{\delta}\right)x^{\frac{1}{2}} - \frac{2\alpha}{3\delta}x^{\frac{3}{2}}.$$

Then, the representation of the option payoff is:

$$E\left[G\left(\{V_t\},\left\{\int_0^t V_s ds\right\}\right)\right]$$

$$= E\left[G\left(\left\{\frac{\delta^2}{4}W_t\right\},\left\{\int_0^t \frac{\delta^2}{4}W_s ds\right\}\right)\exp\left(D(W_T) - D\left(\frac{2}{\delta}\sqrt{V_0}\right)\right.\right.$$

$$\left.\left. -\frac{1}{2}\int_0^T \mu'(W_s)ds - \frac{1}{2}\int_0^T \mu^2(W_s)ds\right)\right]$$

$$= E\left[\left(\frac{S_0}{T}\int_0^T \exp\left\{\left(r - \frac{\rho}{\delta}\alpha\overline{\sigma}^2\right)t + \frac{\rho}{\delta}\left(\frac{\delta^2}{4}W_t - V_0\right) + \left(\frac{\rho}{\delta}\alpha - \frac{1}{2}\right)\frac{\delta^2}{4}\int_0^t W_t ds\right.\right.$$

$$\left.\left. + \sqrt{1-\rho^2}\frac{\delta}{2}\int_0^t \sqrt{W_s}dZ_s\right\}dt - K\right)^{+}\exp\left(D(W_T) - D\left(\frac{2}{\delta}\sqrt{V_0}\right)\right.$$

$$\left. -\frac{1}{2}\int_0^T \mu'(W_s)ds - \frac{1}{2}\int_0^T \mu^2(W_s)ds\right)\Bigg]$$

$$\approx \mathrm{E}\left[\left(\frac{S_0}{L}\sum_{i=1}^{L}\exp\left[\left(r - \frac{\rho}{\delta}\alpha\overline{\sigma}^2\right)t_i + \frac{\rho}{\delta}\left(\frac{\delta^2}{4}W_{t_i} - V_0\right) + \left(\frac{\rho}{\delta}\alpha - \frac{1}{2}\right)\frac{\delta^2}{4}\int_0^{t_i} W_s ds\right.\right.\right.$$

$$\left.\left.\left. + \sqrt{1-\rho^2}\frac{\delta}{2}\int_0^{t_i}\sqrt{W_s}dZ_s\right] - K\right)^{+}\exp\left(D(W_T) - D\left(\frac{2}{\delta}\sqrt{V_0}\right)\right.$$

$$\left. -\frac{1}{2}\int_0^T \mu'(W_s)ds - \frac{1}{2}\int_0^T \mu^2(W_s)ds\right)\Bigg] \qquad (13)$$

`

The BBI-QMC approach with dimension $d = 32$ for pricing the Asian call option has the following steps:

1. A 64-dimensional low discrepancy sequence of length LD is generated so that the BBI-QMC algorithm can be implemented with $dim = 2$ and $d = 32$ in Step 1 through Step 5. Over a given time horizon (i.e. T=1), we have $t_i = \frac{iT}{L}, i = 1,2, \dots, L$. By completion of Step 4, two independent stochastic processes $\left\{W_{t_i} |\, t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$ and $\left\{Z_{t_i} |\, t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$ are simulated with N paths each.
2. Apply the BBD method to calculate LD sets of 16 spot values $\left\{W_{\frac{kT}{16}}, k = 1,2, \dots,16\right\}$ using the first 16 dimensions of the low discrepancy sequence generated in Step 1. Please refer to Step 2 of the BBI-QMC method for details. These spot values together with the initial value W_0 are the end-values, which will be used for the calculation of the conditioning integrals and derivation of the Brownian bridges in Step 3 and Step 4.
3. Given LD sets of spot values $\left\{W_{\frac{kT}{16}}, k = 0,1,2, \dots,16\right\}$, the second 16 dimensions of the low discrepancy sequence are used to generate LD sets of the conditioning integrals

`

$$\left\{X_k = \int_{\frac{k-1}{16}T}^{\frac{k}{16}T} W_s ds\,, k = 1,2,\dots,16\right\}.$$

Thus, for each set, the vector of the 16 conditioning integrals is 16-variate normally distributed with joint distribution

$$[X_1, X_2, \dots, X_{16}] \sim MN\left([m_1, m_2, \dots, m_{16}], \frac{1}{12}\left(\frac{T}{16}\right)^3 I_{16\times 16}\right)$$

where $I_{16\times 16}$ is the identity matrix of size 16 and

$$m_k = W_{(k-1)\frac{T}{16}} + \frac{T}{2}\left[W_{k\frac{T}{16}} - W_{(k-1)\frac{T}{16}}\right], k = 1,2,\dots,16.$$

Note that each conditioning integral is calculated using its two end values and its time duration. Please refer to Step 3 of the BBI-QMC method for details.

4. Conditionally on each set of spot values and conditioning integrals calculated in Step 2 and Step 3, MC Brownian bridges are constructed using BBC method. Thus, a total of $N = LD \times MC$ Brownian paths $\left\{W_{t_i} |\, t_i = \frac{iT}{L}, i = 1,2,\dots,L\right\}$ are generated. Please refer to Step 4 of BBI-QMC method for details.

5. Similar to Steps 2 to 4, a total number of N Brownian paths $\left\{Z_{t_i} |\, t_i = \frac{iT}{L}, i = 1,2,\dots,L\right\}$ are simulated using the BBI-QMC method.

`

6. For each path of $\left\{W_{t_i} \mid t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$, a path of the integrals $\left\{\int_0^{t_i} W_s ds \mid t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$ is calculated.
7. Given function μ in Equation (12), integrals $\int_0^T \mu'(W_s)ds$ and $\int_0^T \mu^2(W_s)ds$ are calculated. Note that both functions μ' and μ^2 are smooth. For functionals of the form $\int_0^T g(s, W_s)ds$, the variability of the integrand is analyzed in Kolkiewicz (2014) when g is a smooth function.
8. For one pair of Brownian paths $\left\{W_{t_i} \mid t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$ and $\left\{Z_{t_i} \mid t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$, the option payoff can be calculated using Equation (13) using the integrals $\left\{\int_0^{t_i} W_s ds \mid t_i = \frac{iT}{L}, i = 1,2, \dots, L\right\}$, $\int_0^T \mu'(W_s)ds$ and $\int_0^T \mu^2(W_s)ds$ calculated in steps 6 and 7.
9. A total of N payoffs are generated by repeating steps 6 to 8.
10. Discount the payoffs by the risk-free rate and take average to get the price of the option as follows:

$$\text{Price} = \frac{1}{N}\sum_{j=1}^{N} e^{-rT}\text{Payoff}^{(j)}.$$

11. Compute the standard error:

$$\text{SE} = \frac{\sqrt{\frac{1}{N-1}\sum_{\text{j=1}}^{\text{N}}(\text{Price} - e^{-rT}\text{Payoff}^{(j)})^2}}{\sqrt{N}}.$$

`

Both accuracy and efficiency of the BBI-QMC method are compared with the crude Monte Carlo method (MC) through different initial settings. The implementation algorithm given above used dimension $d = 32$ for simulation. A comparison using dimensions $d = 8$ and $d = 16$ is applied to investigate the sensitivity of simulation variance to the choice of dimension d. The implementation results are given in Chapter 5.

`

Chapter 5

Implementation Results

The numerical tests in Andersen (2007) are taken as reference for choosing parameters. The speed of mean reversion α is chosen to be 5. The volatility of the volatility process δ is chosen to be 1. The comparison is done in three scenarios when the option is at the money, out of the money and in the money. While the strike price $K = 100$ is fixed, we choose the initial underlying price $S_0 = 100$ in the case of at the money; $S_0 = 90$ for the out of money case and $S_0 = 120$ for the in the money case. Under each scenario, different cases use values 0.1 through 0.5 as the long term volatility mean $\overline{\sigma}$, and the initial volatility V_0 is set to be equal to the long term volatility. The correlation between the two Brownian paths ρ is changing among values $0, -0.3, -0.6$ and -0.9. Other input values are chosen as follows. Time to maturity T is chosen to be 1. The risk free rate r is 0.15. Total time steps L is chosen to be 320. The resulting price and the corresponding standard error (SE) are given together with different parameter settings.

5.1 At the Money

At the money is a situation when we have $S_0 = K$, which in our case is $S_0 = K = 100$. In this scenario, seven different parameter settings are implemented, and the results are provided in Table 2. Note that, for the BBI-QMC method, when the total number

of paths $N = 40000$ is given, we choose $LD = 2000$ and $MC = 20$. When the total number of paths $N = 1000$ is given, we choose $LD = 250$ and $MC = 4$.

Case#	$\bar{\sigma}$	ρ	BBI-QMC			MC		
			N	Price	SE	N	Price	SE
1	0.3	-0.9	40000	10.3053	0.000174	100000	10.3275	0.035041
2	0.4	-0.6	40000	11.8179	0.000495	100000	11.8326	0.049353
3	0.5	-0.6	40000	13.8021	0.000352	100000	13.8506	0.062828
4	0.1	0	1000	7.2212	0.001418	1000	7.2207	0.287521
5	0.2	0	1000	8.8501	0.003153	1000	8.8455	0.379394
6	0.2	-0.3	1000	8.6597	0.002326	1000	8.6343	0.395455
7	0.3	-0.3	1000	10.2329	0.004121	1000	10.1773	0.473055

Table 2: Implementation Results for $S_0 = 100 = K$

As can be seen in Table 2, the large simulation size of the first three cases indicates the accuracy of the BBI-QMC method. The crude Monte Carlo method with $N = 100000$ gives significantly larger standard errors. Comparing with the results from the BBI-QMC method when $N = 40000$, the prices from the BBI-QMC method are covered by the 95% confidence interval of results from the MC method with the same parameter setting. For both the BBI-QMC method and the MC method, the call option price and the corresponding standard error increase as long term volatility mean $\bar{\sigma}$ increases. Cases 4 to 7 are implemented with 1000 paths, which together indicate an overall much smaller standard error of the BBI-QMC method.

`

5.2 Out of the Money

A call option is said to be out of the money when the strike price is higher than the underlying market price, i.e. $S_0 < K$. Table 3 presents results for $S_0 = 90$.

Case#	$\bar{\sigma}$	ρ	BBI-QMC			MC		
			N	Price	SE	N	Price	SE
1	0.2	0	40000	3.03006	0.000062	100000	3.04179	0.023894
2	0.2	-0.3	40000	2.96811	0.000109	100000	2.96509	0.017903
3	0.5	-0.3	40000	8.53310	0.000154	100000	8.52692	0.056185
4	0.1	0	1000	1.89212	0.000713	1000	1.9837	0.143903
5	0.3	-0.6	1000	4.72373	0.001155	1000	4.53732	0.198165
6	0.2	0	1000	3.01909	0.001088	1000	2.8918	0.205618
7	0.4	-0.6	1000	6.50715	0.002302	1000	6.56641	0.307005

Table 3: Implementation Results for $S_0 = 90 < K$

Table 3 above represents the results when the initial underlying price is lower than the strike price, therefore the resulting values in Table 3 are much smaller than those in Table 2. Cases 4 to 7 suggest an overall smaller standard error of results from the BBI-QMC method. Prices calculated by the BBI-QMC method in Cases 1 to 3 are all covered by the 95% confidence interval of results derived by the MC method.

5.3 In the Money

A call option is said to be in the money when the underlying market price is higher than the strike price, i.e. $S_0 > K$. Table 4 presents results for $S_0 = 120$.

`

Case#	$\bar{\sigma}$	ρ	BBI-QMC			MC		
			N	Price	SE	N	Price	SE
1	0.1	-0.9	40000	25.3732	0.000454	100000	25.3831	0.032904
2	0.2	-0.9	40000	25.7203	0.000552	100000	25.6984	0.046225
3	0.3	-0.9	40000	26.2021	0.000657	100000	26.2043	0.064796
4	0.4	-0.9	40000	27.0923	0.001133	100000	27.1535	0.084329
5	0.5	-0.9	40000	28.3039	0.002500	100000	28.3553	0.104472
6	0.1	-0.6	1000	25.3867	0.007231	1000	25.0559	0.313628
7	0.1	-0.3	1000	25.5212	0.009397	1000	25.1186	0.310621
8	0.1	0	1000	25.7520	0.009295	1000	25.2075	0.312209

Table 4: Implementation Results for $S_0 = 120 > K$

In cases 1 to 5 of Table 3, the prices simulated using the BBI-QMC method are again covered by the 95% interval of results from the MC method. The simulation results indicate an overall much smaller standard error of the BBI-QMC method.

5.4 Choices of Dimensions

To investigate the sensitivity of simulation variance to the choice of dimension d, a comparison of the simulation results using different dimensions (i.e. $d = 8$, $d = 16$ and $d = 32$) are conducted. The comparison considers at the money case with $N = 40000$ for BBI-QMC approach and $N = 100000$ for crude MC approach. All parameters are chosen to be the same as Cases 1, 2 and 3 in Section 5.1. The results are presented in Table 5 below.

`

Case#	$\bar{\sigma}$	ρ	SE ($d = 8$)	SE ($d = 16$)	SE ($d = 32$)	MC
1	0.3	-0.9	0.002558	0.000511	0.000174	0.035041
2	0.4	-0.6	0.008952	0.000705	0.000495	0.049353
3	0.5	-0.6	0.011338	0.001213	0.000352	0.062828

Table 5: Choices of Dimensions

It can be seen from the table that by increasing the dimension of the conditioning vector the efficiency of the BBI-QMC approach can be improved. Further, the improvement by changing the dimension from $d = 8$ to $d = 16$ seems more significant than the improvement by changing the dimension from $d = 16$ to $d = 32$.

`

Chapter 6
Conclusion and Future Works

An efficient quasi-Monte Carlo method, developed by Kolkiewicz (2014) , is introduced in detail in this thesis with the objective of pricing an European arithmetic Asian call option under Heston's stochastic volatility model. The key idea of the efficient quasi-Monte Carlo rests on the intelligent use of the low-discrepancy sequence so that Brownian paths are generated conditionally on their integral. The main contribution of this thesis is an extension of Kolkiewicz's method to Asian options under stochastic volatility. Performance of quasi-random sequences and normal inverse algorithms are analyzed in a simulation study.

A special situation has been considered during our implementation. Consider an out of the money call option with $S_0 = 80$ and $K = 100$. Since $d = 2$ is chosen for implementation efficiency, the price path will be simulated conditionally on a random terminal value and a random integral. Figure 14 gives a visual sense of the problem.

`

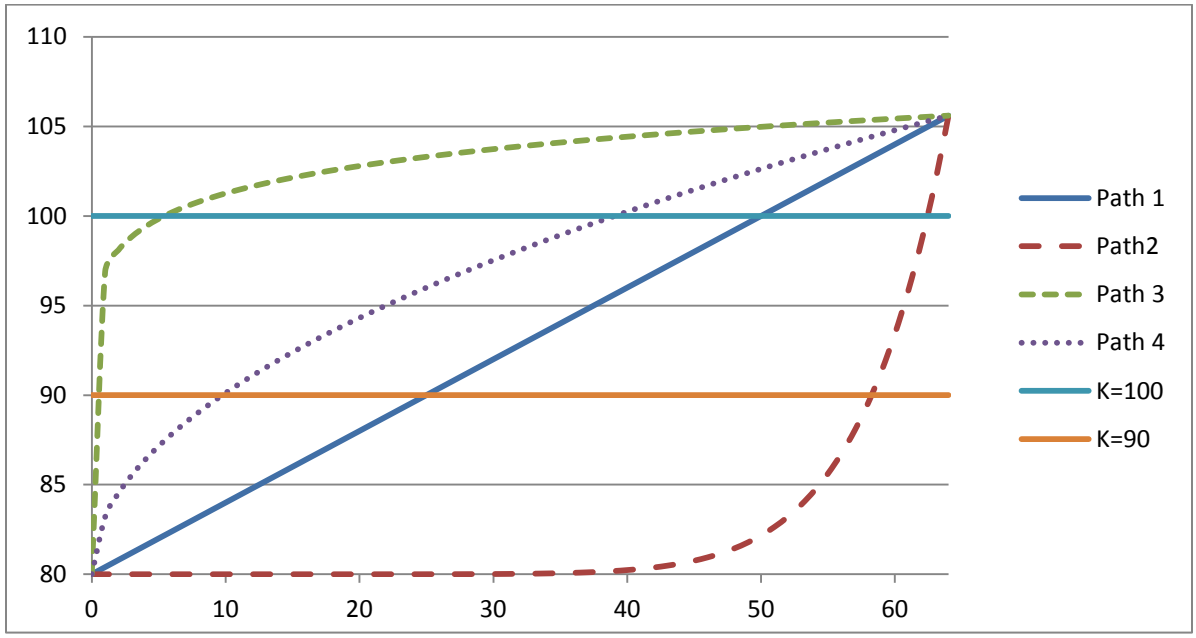

Figure 14: Possible Simulation Error for Out of the Money Call

In Figure 14, there are 4 paths meeting at a terminal value in the top right of the graph. Suppose S_0 is chosen to be significant smaller than K, so that all simulated paths having trend to go below Path 4, such as Path 1 and Path 2 will result in zero payoffs. As introduced in Section 2.4 and Section 4.2, Moro's normal inverse algorithm provides extra efficiency by reducing outliers. Therefore, comparing with MC, BBI-QMC might provide more paths like Path 4 and fewer paths like Path 3. Therefore, in extreme situations when the price of the option heavily relies on outliers, there will be a simulation error, i.e. the call option price will be underestimated. A similar situation for in the money put option will result in over pricing. Figure 15 presents an out of the money put option with $S_0 = 120$ and $K = 100$.

`

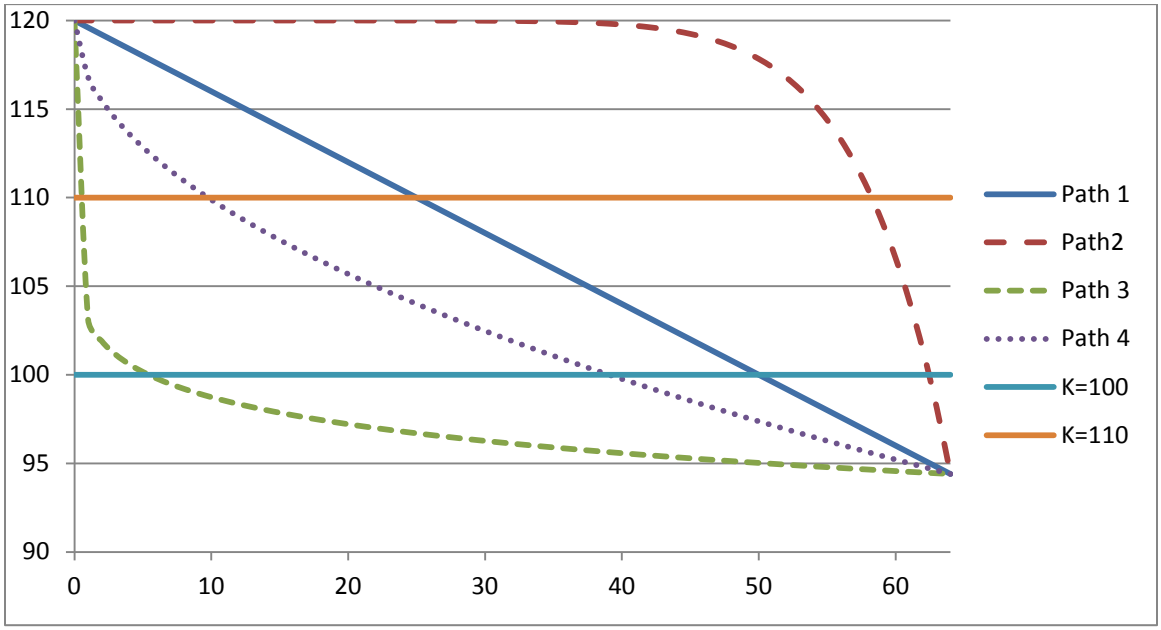

Figure 15: Possible Simulation Error for Out of the Money Put

However, the implementation results do not indicate this kind of error. Analyzing the use of Moro's algorithm for out of the money options can be part of the future works.

In order to use more properly the BBI method, formulation of the problem can be an effective approach. In Kolkiewicz (2014), the BBI method is tested to be very effective in reducing variances when evaluating the following three functionals of Brownian motion $\exp\left(-\int_0^T W(s)ds\right)$, $\exp\left(-\int_0^T W^2(s)ds\right)$ and $\exp\left(-\int_0^T \exp(W(s))ds\right)$. Transformations of the pricing formula, which include more of these functionals, may improve the efficiency of simulation.

`

The implementation results suggest a very good performance in all scenarios of parameter settings. More tests could be done using different interest rate values, volatility's mean reversion speed and time to maturity etc. The Matlab code can be easily applied to problems with higher dimensions. Pricing problems with multiple underlying assets under both stochastic interest rate and stochastic volatility can be applied to further test the efficiency.

Appendix A

Matlab Code for Moro's Normal Inverse Algorithm

```
%This function applies the Beasley-Springer-Moro normal inverse algorithm
%Function Input:
%   p = a probability in [0,1]
%
%Function Output:
%   x such that Phi(x)=p
%
% Note that the Input can be a matirx, but allows only one input


function x=InvNorm(p)

a0=2.50662823884;
a1=-18.61500062529;
a2=41.39119773534;
a3=-25.44106049637;

b0=-8.47351093090;
b1=23.08336743743;
b2=-21.06224101826;
b3=3.13082909833;

c0=0.3374754822726147;
c1=0.9761690190917186;
c2=0.1607979714918209;
c3=0.0276438810333863;
c4=0.0038405729373609;
c5=0.0003951896511919;
c6=0.0000321767881768;
c7=0.0000002888167364;
c8=0.0000003960315187;

u=p-0.5;

index1=abs(u)<0.42;
index2=not(index1);
```

```
x=zeros(size(p));

r1=abs(u(index1));
r=r1.^2;
x(index1)=r1.*polyval([a3,a2,a1,a0],r)./polyval([b3,b2,b1,b0,1],r);

r=p(index2);
r=min(r,1-r);
r=log(-log(r));
x(index2)=polyval([c8,c7,c6,c5,c4,c3,c2,c1,c0],r);

x=x.*sign(u);

end
```

Appendix B

Matlab Code for Quasi-Monte Carlo Method

```
%This function generates Brownian Paths use Efficient Monte Carlo method
%Function Inputs:
%   LD = Length of Low-discrepancy sequences
%   MC = Number of Brownian Bridges to be created for each pair of conditions
%   L = Number of equal spaced points during the whole time period (multiple of d)
%   d = Dimension of the problem, must be a power of 2 (d>=2)
%   T = Time of Maturity
%
%Function Outputs:
%   Bpaths with dimension (LD*MC, L+1)
%      for example an output with dimension (20000, 200) gives
%      20000 simulation results of a brownian motion with
%      length 201


function Bpaths = QMC(LD,MC,L,d,T)
dt=T/L;


%creating Low-discrepancy sequences
%if the overall dimension d*dim is less than 10, we use Haltonset.
%Otherwise Sobolset is used.


if  d*dim > 10
          p = sobolset(d*dim,'Skip',1000,'Leap',100);
          p = scramble(p, 'MatousekAffineOwen');
          Sobol = net(p,LD);
          X = InvNorm(Sobol,0,1);
else
          p = haltonset(d*dim,'Skip',1000,'Leap',100);
          p = scramble(p, 'RR2');
          Halton = net(p,LD);
          X = InvNorm(Halton);
end
```

`

```
%creating the conditional path points


WT = zeros(LD,d+1);
WT(:,d+1)=X(:,d)*sqrt(T);
a=d;
b=1;
while(a>1)
   c=2^b;
   for i=1:2:(c-1)
      WT(:,(i*d/c)+1)=0.5*(WT(:,((i-
1)*d/c)+1)+WT(:,((i+1)*d/c)+1))+0.5*sqrt(T/c)*X(:,i*d/c);
   end
   a=a/2;
   b=b+1;
end


%creating the standard BB paths


BBlength = 1 + L/d;
BM = zeros(d,LD,MC,BBlength);
BB = zeros(d,LD,MC,BBlength);

for i=1:d
   for j=1:LD
      for k=1:MC
         for l=2:BBlength
            BM(i,j,k,l)=BM(i,j,k,l-1)+sqrt(dt)*randn;
         end
         for l=2:BBlength
            BB(i,j,k,l)=BM(i,j,k,l)-BM(i,j,k,BBlength)*(l-1)/(BBlength-1);
         end
      end
   end
end

%creating all paths
```

```
Bpaths = zeros(LD*MC,L+1);

for j=1:LD
   for k=1:MC
      for i=1:d
         for l=1:BBlength
            Bpaths((j-1)*MC+k,(i-1)*(BBlength-1)+l)=WT(j,i) + (l-1)*(WT(j,i+1)-
WT(j,i))/(BBlength-1) + BB(i,j,k,l);
         end
      end
   end
end
```

Appendix C

Matlab Code for BBI-Quasi-Monte Carlo Method

```
%This function generates Brownian Paths use BBI Monte Carlo method
%
%Function Inputs:
%   LD = Length of Low-discrepancy sequences
%   MC = Number of Brownian Bridges to be created for each pair of conditions
%   L = Number of equal spaced points during the whole time period (multiple of d)
%   dim = Number of Independent Stochastic Process to be created
%   d = Dimension of the problem, must be a power of 2 (d>=2)
%   T = Time of Maturity
%
%Function Outputs:
%   Bpaths with dimension (LD*MC, L+1, dim)
%      for example an output with dimension (20000, 201, 2) gives
%      20000 simulation results of 2 independent Brownian motions with
%      length 201


function Bpaths = EfficientQMC(LD,MC,L,dim,d,T)

d0=d/2;
dt=T/L;


%creating Low-discrepancy sequences
%if the overall dimension d*dim is less than 10, we use Haltonset.
%Otherwise Sobolset is used.


if  d*dim > 10
        p = sobolset(d*dim,'Skip',1000,'Leap',100);
        p = scramble(p, 'MatousekAffineOwen');
        Sobol = net(p,LD);
        X = InvNorm(Sobol,0,1);
else
        p = haltonset(d*dim,'Skip',1000,'Leap',100);
        p = scramble(p, 'RR2');
```

```
        Halton = net(p,LD);
        X = InvNorm(Halton);
end


%creating the conditional path points
%this step performs the Brownian Bridge discretization to determine the conditional
path for all #dim processes


WT = zeros(LD,d0+1,dim);
for z=1:dim
   WT(:,d0+1,z)=X(:,z*d0)*sqrt(T);
end

a=d0;
b=1;

while(a>1)
   c=2^b;
   for i=1:2:(c-1)
      for z = 1:dim
         WT(:,(i*d0/c)+1,z)=0.5*(WT(:,((i-
1)*d0/c)+1,z)+WT(:,((i+1)*d0/c)+1,z))+0.5*sqrt(T/c)*X(:,i*d0/c+((z-1)*d0));
      end
   end
   a=a/2;
   b=b+1;
end


%creating the conditional integrals
%there are d0 integrals for each single path


MU = zeros(LD,d0,dim);
SIG = sqrt((T/d0)^3/12);
A = zeros(LD,d0,dim);

for i=1:d0
```

```
   for z=1:dim
      MU(:,i,z)=0.5*(WT(:,i,z)+WT(:,i+1,z))*T/d0;
      A(:,i,z)=MU(:,i,z)+SIG*X(:,d0*dim+(z-1)*d0+i);
   end
end


%creating the standard Brownian Bridge paths
%together with the conditional terminal values and integrals from last two steps, the
complete paths can be generated


BBlength = 1 + L/d0;
BM = zeros(d0,LD,MC,BBlength,dim);
BB = zeros(d0,LD,MC,BBlength,dim);
IntBB = zeros(d0,LD,MC,dim);

for i=1:d0
   for j=1:LD
      for k=1:MC
         for l=2:BBlength
            for z=1:dim
               BM(i,j,k,l,z)=BM(i,j,k,l-1,z)+sqrt(dt)*randn;
            end
         end
         for l=2:BBlength
            for z=1:dim
               BB(i,j,k,l,z)=BM(i,j,k,l,z)-BM(i,j,k,BBlength,z)*(l-1)/(BBlength-1);
               IntBB(i,j,k,z)=IntBB(i,j,k,z)+BB(i,j,k,l,z)*dt;
            end
         end
      end
   end
end


%creating all paths
%this is the final step generating #dim processes with length #L+1 and simulation
time #LD*MC
```

```
Bpaths = zeros(LD*MC,L+1,dim);

for j=1:LD
   for k=1:MC
      for i=1:d0
         for l=1:BBlength
            for z=1:dim
               Bpaths((j-1)*MC+k,(i-1)*(BBlength-1)+l,z)=WT(j,i,z) + (l-1)*(WT(j,i+1,z)-
WT(j,i,z))/(BBlength-1) + BB(i,j,k,l,z) - (6*(l-1)*(BBlength-l)/(dt*((BBlength-
1)^3)))*(IntBB(i,j,k,z)-A(j,i,z)+0.5*(BBlength-1)*dt*(WT(j,i,z)+WT(j,i+1,z)));
            end
         end
      end
   end
end
```

Bibliography

Acworth, P., Broadie, M., Glasserman, P. et al. (1996) "A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing" *Monte Carlo and Quasi-Monte Carlo Methods,* Springer-Verlag, New York, 1-18

Andersen, L. (2007) "Efficient Simulation of the Heston Stochastic Volatility Model" Bank of America Securities, Available at SSRN: http://ssrn.com/abstract=946405

Antonov, I.A. and Saleev, V.M. (1997) "An economic method of computing LP_τ-sequences" Zh. Vych. Mat. Fiz., 19: 243-245

Boyle, P.,(1977) "Options: A Monte Carlo Approach", Journal of Financial Economics,4, 323-338

Beasley J.D.; Springer S.G.; Wilmot P. (1977) "The Percentage Points of the Normal Distribution", Applied Statistics, 26, 118-121

Berkaoui, A., Bossy, M., and Diop, A. (2008) "Euler scheme for SDEs with non-Lipschitz diffusion coefficient: strong convergence" Probability and Statistics, 12, 1-11

Caflisch, R.E. and Moskowitz, B. (1995) "Modified Monte Carlo methods using quasi-random sequences" H. Niederreiter and P.J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Compting*, Lecture Notes in Statistics, Springer-Verlag, 106: 1-16

Caflish, R.E., Morokoff, W. and A. Owen (1997) "Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension", Journal of Computational Finance 10: 129-155

Cox, J.C., Ingersoll, J.E. and Ross, S.A. (1985) "A theory of the term structure of interest rates" Econometrica, 53: 385-407

Faure, H. (1982) "Discrepance de suites associees a un systeme de numeration (en dimension s)", Acta Arithmetta, 41, 337-351

Forsyth, P.A., Vetzal, K.R. and Zvan, R. (1998) "Robust Numerical Methods for PDE Models of Asian Options", Journal of Computational Finance, 1, 38-47

Glasserman, P. (2003) *Monte Carlo Methods in Financial Engineerin,* Applications of Mathematics, Springer

Halton, J. (1960) "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals", *Numerische Mathematik*, vol. 2, no.1

Halton, J. (1964) "Algorithm 247: Radical-inverse quasi-random point sequence", Association for Computing Machinery, 7, 701-702

Heston, S. (1993) "A closed-form solution for options with stochastic volatility with applications to bond and currency options", The Review of Financial Studies, 6, 327-343

Hlawka, E. (1961) "Funktionen von beschrankter variation in der theorie der gleichverteilung" Annali di Matematica Pura ed Applicata, 54: 325-333

Hull, J.; White, A. (1987) "The Pricing of Options on Assets with Stochastic Volatility", Journal of Finance, 42, 281-300

Imai, J. and Tan, K.S. (2006) A general dimension reduction method for derivative pricing, Journal of Computational Finance, 10: 129-155

Keifer, J. (1961) "On large deviation of the empiric d.f. of vector chance variables and a law of the iterated algorithm" Pacific Journal Math, 11:649-660

Koksma, J.F. (1943) "Een algemeene stelling uit de theorie der gelikmatige verdelling modulol", Mathematica B (Zutphen), 11: 7-11

Kuipers L. and Niederreiter H. (1974) *Uniform Distribution of Sequences*, John Wiley & Sons, New York

Kolkiewicz, A. (2014) "Efficient Monte Carlo simulation for integral functionals of Brownian motion", Journal of Complexity, 30, 255-278

Moro, B. (1995) "The Full Monte", Risk, 8(2), 57-58

Morokoff, W.J., "Generating quasi-random paths for stochastic process", Society for Industrial and Applied Mathematics Review 40: 765-788

Niederreiter, H. (1992) "Random Number Generation and Quasi-Monte Carlo Methods", Society for Industrial and Applied Mathematics, Philadelphia

Paskov, S. and Traub J. (1995) "Faster valuation of financial derivatives", Journal of Portfolio Management 22:113-120

Paskov, S. (1997) "New Methodologies for Valuing Derivatives", *Mathematics of Derivative Securities*, Cambridge University Press, Cambridge, 545-582

Paskov, S. (1998) "New Methodologies for Valuing Derivatives", *Monte Carlo: Methodologies and Applications for Pricing and Riak Management*, 281-298

Press, W.H; Flannery, B.P.; Teukolsky, S.A. and Vetterling, W.T. (1992) *Numerical Recipes in C: The Art of Scientific Computing, 2nd ed.* Cambridge University Press

Sobol, I.M. (1967) "On the distribution of points in a cube and the approximate evaluation of integrals", Computational Mathematics and Mathematics Physics, 7, 86-112

Tezuka S. (1998) "Financial Applications of Monte Carlo and Quasi-Monte Carlo Methods", Random and Quasi-Random Point Sets, P. Hellekalek et al. (eds.), Springer-Verlag New York, 303-332

`

Vecer, J. (2001) “New Pricing of Asian Options” Preprint, Columbie University, available online at *http://www.infres.enst.fr/~decreuse/TP/FE-KyotoU_WP01.pdf*

Wang, X. and Tan K.S. (2012) "How do path generation methods affect the accuracy of quasi-Monte Carlo methods of problems in finance?" Journal of Complexity, 28:250-277