

**EVALUATING OF PATH-DEPENDENT SECURITIES WITH
LOW DISCREPANCY METHODS**

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Professional Degree of Master of Science

in

Financial Mathematics

by

Inna Krykova

December 2003

APPROVED:

Professor Domokos Vermes, Major Advisor

Professor Bogdan Vernescu, Head of Department

ABSTRACT

The objective of this thesis is the implementation of Monte Carlo and quasi-Monte Carlo methods for the valuation of financial derivatives. Advantages and disadvantages of each method are stated based on both the literature and on independent computational experiments by the author. Various methods to generate pseudo-random and quasi-random sequences are implemented in a computationally uniform way to enable objective comparisons.

Code is developed in VBA and C++, with the C++ code converted to a COM object to make it callable from Microsoft Excel and Matlab. From the simulated random sequences Brownian motion paths are built using various constructions and variance-reduction techniques including Brownian Bridge and Latin hypercube. The power and efficiency of the methods is compared on four financial securities pricing problems: European options, Asian options, barrier options and mortgage-backed securities. In this paper a detailed step-by-step algorithm is given for each method (construction of pseudo- and quasi-random sequences, Brownian motion paths for some stochastic processes, variance- and dimension- reduction techniques, evaluation of some financial securities using different variance-reduction techniques etc).

ACKNOWLEDGEMENTS

I would like to express my warm thanks to my thesis advisor, Domokos Vermes, for his help and support. Prof. Vermes helped me to chose a very interesting and important topic in Finance for investigation and make progress toward the completion of my work. With his encouragement, I was able to overpass all the difficulties in my research and complete this project successfully.

I also want to thank all Mathematical Sciences Department and Management Department, all the faculty and staff who had taught me during this program and advised me.

I would like to thank Prof. Ann Wiedie for her advices and help in finding an Actuarial Summer Internship position that, fortunately, has led to a full-time employment.

I also would like to thank all the students of Financial Math for their guidance and inspiration me with financial applications.

I also want to thank my family for moral, financial support and for their patience during my MS's study. Special thanks to my husband who helped me with COM objects.

I am very thankful to Professor Vermes for his knowledge and critical guidance, for the endless hours he spent reviewing and editing the drafts of my thesis, and for his kindness in accommodating my schedule.

CONTENTS

1 INTRODUCTION	5
2 LOW DISCREPANCY INTEGRATION METHODS	7
2.1 Monte Carlo.....	7
2.2 Quasi-Monte Carlo.....	9
2.3 Random Uniform Sequences	11
2.4 Low Discrepancy Sequences.....	12
2.4.1 Halton Sequences.....	13
2.4.2 Faure Sequences.....	17
2.4.3 Sobol Sequences.....	20
2.4.4 A Hybrid Quasi-Monte Carlo Approach	23
2.4.5 (t,m,s) -NETS and (t,s) -SEQUENCES.....	26
2.4.6 Comparison Low Discrepancy Sequences	27
2.5 Generating of Normally Distributed Sequences	30
3 GENERATION OF MATRICES OF BROWNIAN MOTION (BM) PATHS	34
3.1 Generation BM paths using Forward Increments.....	35
3.1.1 Generation Paths for general stochastic process	35
3.1.2 Generation Paths for Geometric Brownian motion	35
3.1.3 Generation Paths for Geometric Brownian motion of multiple correlated assets	36
3.1.4 Generation Paths of Term Structure models	37
3.2 Generation BM paths using some variance-reduction techniques	39
3.2.1 Antithetic variates	39
3.2.2 Control variates	40
3.2.3 Importance Sampling	42
3.2.4 Stratified sampling	43
3.2.5 Latin Hypercube Sampling	44
3.2.6 Generation BM paths using Brownian Bridge.....	45
3.3 Demonstration of BM paths with different variance-reduction techniques	46
4 APPLICATIONS TO FINANCE PROBLEMS	49
4.1 Evaluating European options	50
4.2 Evaluating Barrier Options	53
4.3 Evaluating Asian Options	55
4.4 Evaluating Mortgage with Prepayment Options	57
5 CONCLUSIONS	61
BIBLIOGRAPHY	64

CHAPTER 1

INTRODUCTION

To price financial derivatives analytic methods are only available for a few special cases and often under unrealistically restricting assumptions. In most practical cases computational methods have to be used.

Derivatives whose price depends only on the current value of the underlying security can be priced using partial differential equations. For numerical evaluation these equations can be discretised and solved e.g. by finite differences. Current technology allows the solution of the partial differential equations up to 6-7 dimensions.

Many important derivatives are either higher dimensional or are path dependent, which means that their prices depend on the past history of the value of the underlying security, not just on its current value. Important such examples are Asian options, which depend on the average price of the underlying over a certain interval of time. Another example is mortgage-backed securities, whose price depends on the past evolution of interest rates. Such path-dependent securities cannot be valued using partial differential equations. In these cases Monte Carlo simulation is the most efficient computational method to evaluate them.

The price of a derivative security is always a discounted expected value with respect to a risk-neutral martingale measure. Due to the extreme complexity of the random variables involved, this expected value can only be evaluated by numerical integration. In Monte Carlo methods the random variables are simulated by computer generated pseudo-random sequences and the numerical integration is performed by averaging over a large number of simulations. Pseudo-random numbers mimic the realizations of independent identically distributed random variables. Due to a fundamental result in probability theory, Monte Carlo methods based on such pseudo-random numbers can converge only proportionally to the square root of the number of simulations.

It was noticed that the randomness of the simulating sequence is not essential for numerical integration. Deterministic sequences which fill the space uniformly can also be used. These so called low-discrepancy sequences offer the advantage of a faster convergence proportional to the number of simulations (and not the square root of it). Simulation methods based on low-discrepancy sequences are also called quasi-Monte Carlo techniques.

Realistic problems in computational finance are typically high dimensional. For example pricing a mortgage backed security requires a 360 dimensional space. Solution of such high dimensional problems is currently limited by the available computer technology. Consequently, using more efficient numerical methods makes it possible to solve problems whose solution wouldn't be possible otherwise with the current technology. Quasi-Monte Carlo techniques are considered a promising new methodology for financial problems.

The goal of the present thesis is the implementation of Monte Carlo and quasi-Monte Carlo methods for the valuation of financial derivatives. Advantages and disadvantages of each method are stated based on both the literature and on independent computational experiments by the author. Various methods to generate pseudo-random and quasi-random sequences are implemented in a computationally uniform way to enable objective comparisons. In the literature most authors state that their methods are superior to others. This issue is further complicated by the fact, that some of the highest publicized methods are only available as proprietary commercial implementations. Their

authors do not disclose crucial details and, at the same time, claim unsurpassed superiority. To even the playing field and to make objective comparisons possible, all methods considered in this thesis were coded in a uniform way by the author. As a consequence, observed processor times are indicators of the efficiency of the methods and are not influenced by their computational implementations. From the simulated random sequences Brownian motion paths are built using various constructions and variance-reduction techniques including Brownian Bridge and Latin hypercube. The power and efficiency of the methods is compared on four financial securities pricing problems: European options, Asian options, barrier options and mortgage-backed securities. In this paper a detailed step-by-step algorithm is given for each method (construction of pseudo- and quasi-random sequences, Brownian motion paths for some stochastic processes, variance- and dimension- reduction techniques, evaluation of some financial securities using different variance-reduction techniques etc).

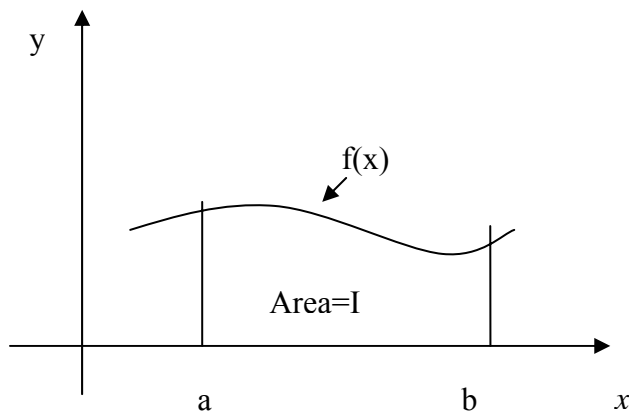
CHAPTER 2

LOW DISCREPANCY INTEGRATION METHODS

2.1 Traditional Monte Carlo (MC) integration

Monte Carlo integration is to use random points for the numerical evaluation of an integral. Other words, to use random points to determine the area under the function, see the picture below (based in [Press](#) et al., 1992).

Let consider a simple example. We want to evaluate an area on a picture below.



According to numerical methods to find the area (or evaluate the integral) we can approximately

compute it as a sum:
$$I = \int_a^b f(x) dx \approx \frac{(b-a)}{N} \sum_{i=1}^N f(x_i).$$

Monte Carlo estimators will approximate I by taking a lot of random samples x_i and averaging their contributions $f(x_i)$.

The Monte Carlo simulation can be viewed as a problem of integral evaluation. Recall that to calculate an expected value we have to evaluate an integral (or a summation for discrete probability distributions). [Ripley](#) wrote that the object of any simulation study is the estimation of one or more expectations of the form $E[\varphi(X)]$. In general, the Monte Carlo method solves multidimensional integrals, and the expression for the Monte Carlo approximation for the multidimensional integral is given by (1).

$$(1) \quad \theta = \int_B \varphi(x) f(x) dx \approx \frac{1}{N} \sum_{\substack{i=1 \\ x_i \in B}}^N \varphi(x_i),$$

where $x_i, i = \overline{1, N}$, are N independent random samples from the distribution of $f(x)$ that are obtained from N independent random samples from uniform distribution on $[0,1]^s$ (s is the dimension of the unit cube, $B \in \mathfrak{R}^s$).

The Monte Carlo method provides approximate solutions to a variety of mathematical problems by performing statistical sampling experiments on a computer. Monte Carlo methods have been used for centuries, but only in the past several decades has the technique gained the status of a full-fledged numerical method capable of addressing the most complex applications. The method applies to problems with no probabilistic content as well as to those with inherent probabilistic structure. In the computational practice of MC methods the required random numbers and random vectors are actually generated by the computer in a deterministic subroutine. In this case are *pseudorandom numbers* and *pseudorandom vectors* are meant.

Advantages of MC. Because of its intuitive sense and simplicity of implementation Monte Carlo methods are used in a wide range of applications. The advantages of MC are:

- This method is easy to implement. There are only minimal requirements that make the method applicable to very difficult integration problems. The evaluation requires only the ability to *sample random points* x and evaluate $f(x)$ for these points.
- MC method for numerical integration offers a way to deal with problems of high dimensionality.
- The standard error does not depend upon the dimensionality of the integral whereas most techniques of numerical integration—such as the trapezoidal rule or Simpson's method—suffer from the *curse of dimensionality*.
- Monte Carlo simulation allows mimicking extremely complicated phenomena by using the computer's random numbers generator to simulate normal fluctuations in nature.

Limitations of MC. Although of all above strength sides, Monte Carlo method is not a panacea. It has several deficiencies that may complicate its usefulness [[Niederreiter](#)]. Some of the disadvantages are:

- It must be decided in advance how many points to choose and how fine it is should be. Once number of random points is chosen all those sample points should be completing. With a grid it is not convenient to “sample until” convergence or termination criterion is met. So, with MC methods, generating random samples is difficult.
- The MC method converges slowly. The *convergence rate* —how quickly the error decreases with the number of samples—of basic Monte Carlo integration is proportional to $\frac{1}{\sqrt{N}}$. This means that to halve the error, four times as many samples are needed.
- The results are statistical in nature. This means that the estimate I can be wrong, and there are only probabilistic error bounds.
- The error bound does not reflect any additional regularity of the integrand.
- The Monte Carlo method depends on our initial seed.

Mechanism of MC integration. To evaluate integrals in form (1) and their errors using Monte Carlo Method we can implement follow algorithm:

1. Generate a sequence (x_1, x_2, \dots, x_N) of random numbers with density function $f(x)$.
2. Form a sum $\hat{\theta}_N = \frac{1}{N} \sum_{i=1}^N \varphi(x_i)$ that is the approximation for $\theta = \int_B \varphi(x) f(x) dx$.
3. Evaluate the empirical variance of θ : $S_N^2 \approx \frac{1}{N-1} \sum_{i=1}^N (\varphi(x_i) - \hat{\theta}_N)^2$.
4. Compute the standard error (SE): $SE(\hat{\theta}_N) = \frac{\sqrt{S_N^2}}{N}$.

SE, a measure of the error, is the standard deviation of $\hat{\theta}_N$. The error is due the fact that it is an average of randomly generated samples and so itself random.

5. Calculate the necessary sample size N_0 to achieve error less or equal to ε , where $\varepsilon_\alpha = \frac{2S_N}{\sqrt{N}}$. From the above with $(1 - \alpha)\%$ confidence we have

$$P\left(\left|\hat{\theta}_N - \theta\right| < \frac{2S_N}{\sqrt{N}}\right) \geq (1 - \alpha)$$

The Monte Carlo method was invented by Stanislaw Ulam, a Polish born mathematician who worked for John von Neumann on the United States' Manhattan Project during World War II, in 1946 while pondering the probabilities of winning a card game of solitaire. But Ulam did not invent statistical sampling. This had been employed to solve quantitative problems before, with physical processes such as dice tosses or card draws being used to generate samples. W. S. Gossett, who published under the pen name "Student," randomly sampled from height and middle finger measurements of 3,000 criminals to simulate two correlated normal distributions. Ulam's contribution was to recognize the potential for the newly invented electronic computer to automate such sampling. Working with John von Neuman and Nicholas Metropolis, he developed algorithms for computer implementations, as well as exploring means of transforming non-random problems into random forms that would facilitate their solution via statistical sampling. This work transformed statistical sampling from a mathematical curiosity to a formal methodology applicable to a wide variety of problems. Metropolis named the new methodology after the casinos of Monte Carlo. The Monte Carlo method (stochastic simulation) was introduced in finance in 1977, in the pioneering work of [Boyle](#).

2.2 Quasi-Monte Carlo

Quasi-Monte Carlo integration is a method of numerical integration that operates in the same way as Monte Carlo Integration, but instead uses sequences of quasi-random numbers which have a more uniform behavior to compute the integral. Quasi-random numbers are generated algorithmically by computer, and are similar to pseudo-random numbers while having the additional important property of being deterministically chosen based on equally distributed

sequences (*Ueberhuber* 1997, p. 125) in order to minimize errors. In general this change will cause the integration estimate to converge towards the actual solution like $(\ln N)^s / N$ (where s is the number of dimensions in the integral) instead of the usual $\frac{1}{\sqrt{N}}$ of the standard MC procedure.

This improved convergence is considerably better, almost as fast as $\frac{1}{N}$. The term “quasi-random” is somewhat misleading because is nothing “random”. The sample points in a quasi-random sequence are, in a precise sense, “maximally avoiding” of each other.

QMC methods can be viewed as deterministic versions of Monte Carlo methods [[Niederreiter](#)]. Determinism enters in two ways: 1) by working with deterministic points rather than random samples and 2) by the availability of deterministic error bounds instead of probabilistic MC error bounds. It could be argued that most practical implementations of MC methods are, in fact, quasi-Monte Carlo methods since the purportedly random samples that are used in Monte Carlo calculation are often generated in the computer by the deterministic algorithm. In QMC methods deterministic nodes are selected in such a way that the error bound is as small as possible. The very nature of the QMC methods with its completely deterministic procedures implies that we get deterministic and thus guaranteed error bounds [[Niederreiter](#)]. In principle, it is therefore always possible to determine in advance an integration rule that yields a given accuracy. For example, for MC it is necessary to increase 100 times the number of simulations N to reduce the error by a factor of 10, whereas the QMC requires less (in general much less) than 100 times, and only 10 times in optimal cases.

Quasi Monte Carlo (QMC) methods are also termed *low discrepancy* procedures.

The discrepancy of the point set $\{x_i\}_{i=1, \overline{N}} \in [0,1)^s$ is $D_N^{(s)} = \sup_E \left| \frac{A(E; N)}{N} - \lambda(E) \right|$

where $E = [0, t_1) \times \dots \times [0, t_s)$, $0 \leq t_j \leq 1$, $j = \overline{1, s}$, $\lambda(E)$ the Lebesgue measure of E is, and $A(E; N)$ is the number of x_i contained in E .

Other words, E is an s -dimensional hyper-rectangle, $\lambda(E)$ is the length, area, volume etc, $\frac{A(E; N)}{N}$ is the percentage of x_i in E , and the discrepancy is the largest difference between $\frac{A(E; N)}{N}$ and $\lambda(E)$.

A **Low Discrepancy Sequence** is a set of s -dimensional points, filling the sample area “efficiently” and has a lower discrepancy than straight pseudo-random number set. The advantages of QMC:

- The integral is approximated using a well-chosen sequence of points.
- The QMC approach often leads to better point estimates for a similar computational effort compared with standard Monte Carlo.
- Quasi-random numbers result in faster convergence. Fewer quasi-random samples are needed to achieve a similar level of accuracy as obtained by pseudo-random sequences.

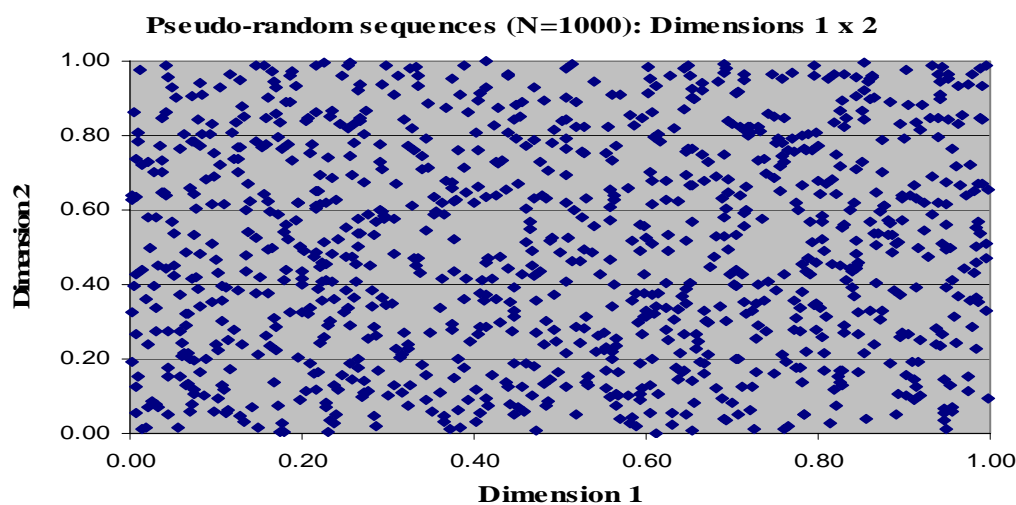
- In several cases permit to improve the performance of Monte Carlo simulations, offering shorter computational times and/or higher accuracy. Because of higher accuracy, Quasi-Monte Carlo methods provide a way to improve the accuracy and reliability of Monte Carlo simulation.

Quasi-Monte Carlo methods have first been proposed in the 1950s and their theory have since developed widely. Some of applications of QMC methods arise in problems of numerical analysis that can be reduced to numerical integration. In 1995 [Paskov and Traub](#) used quasi-Monte Carlo (QMC) methods to estimate the price of a collateralized mortgage obligation. The problem they consider was in high dimensions (360) but nevertheless, they obtained more accurate approximations with QMC methods than with the standard MC method. Since then, many people have been looking at QMC methods has a promising alternative for pricing financial products. In finance applications it turns out, that results obtained by QMC are usually closer to the best rate of convergence than the theoretic worst-case. This occurs sometimes because the use of some technique for the *reduction of effective dimension*, sometimes because the favorable *smooth payoff function* and even the discount effect [[Boyle](#)].

2.3 Random Uniform Sequences (pseudo-random)

The *pseudo-random* sequence of numbers looks like random numbers because it looks unpredictable. However, pseudo random numbers are generated with *deterministic* (explaining the adjective "pseudo") algorithm like the *congruent random generator* (the most common generator). In addition, the implementation of these pseudo random sequences are, in general, of *volatile type* in the sense that the *seed* (initial value of a sequence) depends of an (unpredictable) external feeder like the computer clock.

The Uniform distribution in the interval $[0, 1]$ is, for practical purposes, the only distribution that we need generate for simulations. The reason is that the samples from the other distributions are derived using the Uniform Distribution.



There is no need to show pseudo-random sequences for other dimensions because they will look similar to the above graph thanks to randomness.

2.4. Low Discrepancy (LD) Sequences

Discrepancy is a measure of deviation from uniformity of a sequence of points in D ($D = [0,1]^s$). Discrepancy contributes to the error in quasi-Monte Carlo methods. This error is bounded by the product of a function's variation, a measure of the function's "niceness" over the area of integration, and the discrepancy of the point set. The idea behind the low-discrepancy sequences (a sequence of n -tuples that fills n -space more uniformly than uncorrelated random points) is that for any rectangular set B the fraction of the points within B should be as "close" as possible to its volume. That way (see [Press et al.](#) for detail), the low-discrepancy sequences cover the unit cube as "uniformly" as possible by reducing gaps and clustering of points [[Paskov](#)]. Although the ordinary uniform random numbers and quasi random sequences both produce uniformly distributed sequences, they are a very different. A uniform random generator on $[0,1)$ will produce outputs so that each trial has the same probability of generating a point on equal subintervals, for example $[0,1/2)$ and $[1/2,1)$. Therefore, it is possible for n trials to coincidentally all lie in the first half of the interval, while the $(n + 1)^{\text{st}}$ point still falls within the other of the two halves with probability $1/2$. This is not the case with the quasi random sequences, in which the outputs are constrained by a low-discrepancy requirement that has a net effect of points being generated in a highly correlated manner (i.e., the next point "knows" where the previous points are). Such a sequence is extremely useful in computational problems where numbers are computed on a grid, but it is not known in advance how fine the grid must be to obtain accurate results. Using a quasi random sequence allows stopping at any point where convergence is observed, whereas the usual approach of halving the interval between subsequent computations requires a huge number of computations between stopping points.

The van der Corput sequence is the simplest one dimensional low discrepancy sequence. To obtain n^{th} point x_n of the van der Corput sequence (with a prime base of p), first write the integer n

in base p : $n = \sum_{i=0}^I a_i(n) * p^i$, then transpose the digits $a_i(n)$ around the "decimal point" to get

the corresponding quasi-random number $x_n = \Phi_p(n) = \sum_{i=0}^I \frac{a_i(n)}{p^{i+1}}$.

Only a finite number of these $a_i(n)$ will be non-zero. I is the lowest integer that makes $a_i(n) = 0$ for all $i > I$ (I equal to integer part of $\ln(n) / \ln(p)$, $n = \overline{1, N}$).

For example, let $p=3$ and $n=19$. We can write 19 in base 3 as $19 = 2 * 3^2 + 0 * 3^1 + 1 * 3^0 = 201$. When reflecting 201 (in base 3) about the "decimal point" we obtain $x_{19} = \Phi_3(19) = \frac{1}{3} + \frac{0}{9} + \frac{2}{27} = \frac{11}{27}$. This is a number in the interval $[0,1]$.

The sequence with a base of 2 begins:

$$n=1: 1 = 1 * 2^0 = 1, x_1 = \Phi_2(1) = \frac{1}{2};$$

$$n=2: 2 = 1 * 2^1 + 0 * 2^0 = 2, x_2 = \Phi_2(2) = \frac{0}{2} + \frac{1}{4} = \frac{1}{4};$$

$$n=3: 3 = 1 * 2^1 + 1 * 2^0 = 11, x_3 = \Phi_2(3) = \frac{1}{2} + \frac{1}{4} = \frac{3}{4};$$

$$n=4: 4 = 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 100, x_4 = \Phi_2(4) = \frac{0}{2} + \frac{0}{4} + \frac{1}{8} = \frac{1}{8} \dots$$

After every $N=2^n-1$ points, the sequence is “maximally spread out”, i.e. the longest interval $(a, b) \subseteq (0,1)$ which does not contain any points from the sequence is as short as possible. The construction process of new LD sequences involves sub-dividing the unit hypercube into sub-volumes (boxes) of constant volume, which have faces parallel to the hypercube's faces. The idea is to put a number in each of these sub-volumes before going to a finer grid.

Different bases have different *cycle length*, the quantity of numbers to cover the interval $[0, 1)$ in each cycle. In base two, the pairs $(0, 1/2)$ and the pair $(1/4, 3/4)$ are the two first cycles. For other bases this cycle is larger. For example, in base 3 the sequence has power of 3 denominator with length cycle = 3 (e.g., $0, 1/3, 2/3$ is the first cycle). It looks like:

$$0, 1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9, 1/27, 10/27, 19/27 \dots$$

Two keys to the successful use of QMC methods in high dimensions are the construction of good sequences and the intelligent use of the sequences for path generation. There are several high-dimensional sequences for use in QMC: Halton, Faure, and Sobol sequences. Other important sequences are Niederreiter [[Niederreiter](#)] and generalized Faure sequences (see [[Bratley](#)], [[Ninomiya](#)]).

2.4.1 Halton Sequences

Halton sequence is the most basic low discrepancy sequence in multiple dimensions, which can be viewed as the *building block* of other low discrepancy sequences. The Halton sequence [[Broadie](#)] is a general s -dimensional sequence in the unit hypercube $[0,1]^s$. The first dimension of the Halton sequence is the van der Corput sequence base 2 and the second dimension is the van der Corput sequence using base 3. Dimension s of the Halton sequence is the van der Corput sequence using the s -th prime number as the base. As the base of the van der Corput sequence is getting larger as the dimension increases, it takes increasingly longer to fill the unit hypercube (for example, 25th and 26th primes are 97 and 101 correspondingly). The sequence corresponding to the prime p has cycles of length p with numbers monotonically increasing. This characteristic makes the initial terms of two sequences highly correlated, at least to the first cycle of each sequence.

In one dimension for a prime base p , the n th number in the sequence $\{H_n\}, n = \overline{1, N}$, is obtained by the following steps [[Press at al.](#)].

For each $n = \overline{1, N}$:

1. Write n as a number in base p . For example, suppose $p = 3$ and $n=22$, then we can write 22 in base 3 as $22 = 2 * 3^2 + 1 * 3^1 + 1 * 3^0 = 211$.
2. Reverse the digits and put a radix point (i.e. a decimal point base p) in front of the sequence (in the example, we get 0.112 base 3).

3. The result is H_n .

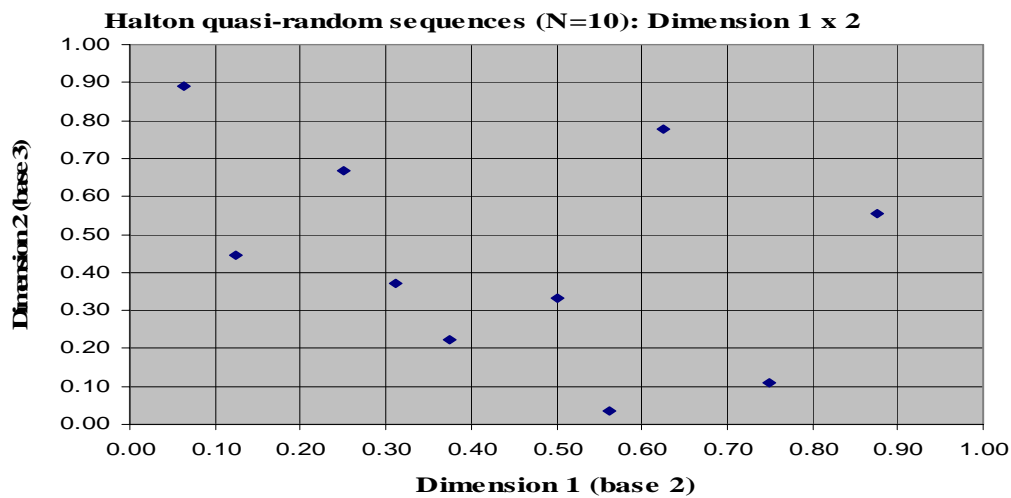
In s -dimension problem, each component a Halton sequence are made with a different prime base p (first n primes are used). Every time the number of digits in n increases by one place, n 's digit-reserved fraction becomes a factor of p finer-meshed. So, at each step as n increases points of Halton sequence are better and better filling Cartesian grids.

Table 1. Halton sequences for first 3 dimensions

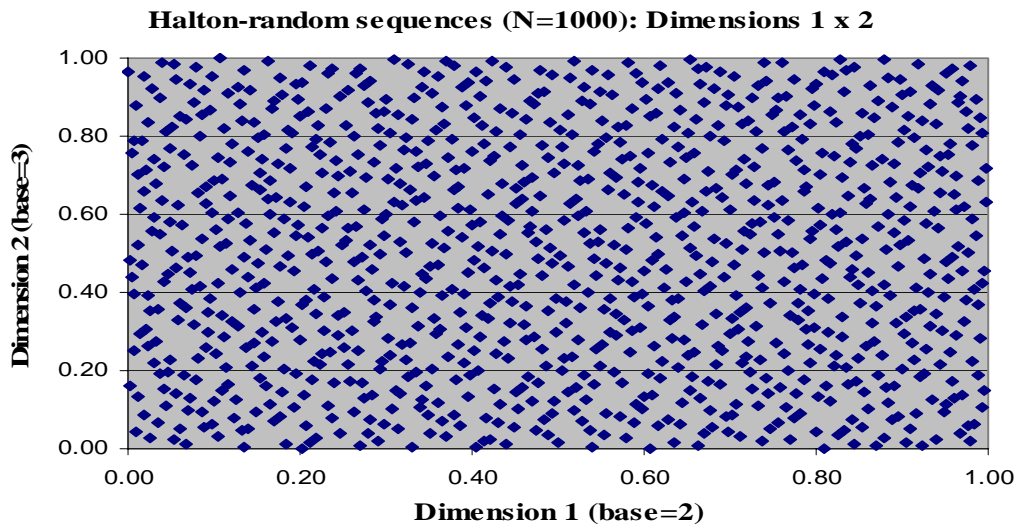
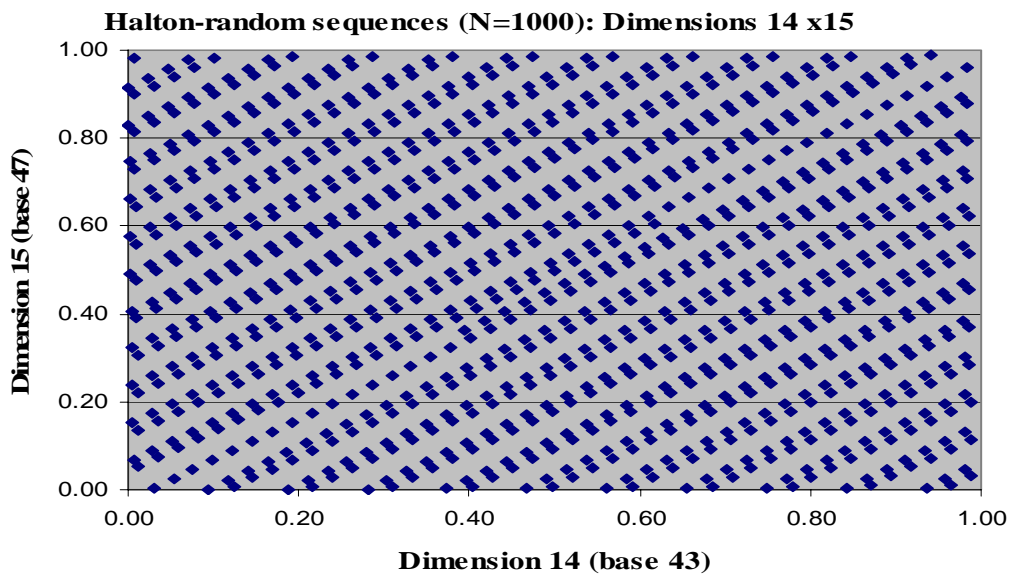
	<i>Dim=1</i> <i>(Base 2)</i>	<i>Dim=2</i> <i>(Base 3)</i>	<i>Dim=3</i> <i>(Base 5)</i>
<i>n=1</i>	1/2	1/3	1/5
<i>n=2</i>	1/4	2/3	2/5
<i>n=3</i>	3/4	1/9	3/5
<i>n=4</i>	1/8	4/9	4/5
<i>n=5</i>	5/8	7/9	1/25
<i>n=6</i>	3/8	2/9	6/25
<i>n=7</i>	7/8	5/9	11/25
<i>n=8</i>	1/16	8/9	16/25

Figure 1 below shows first 10 quasi-random numbers of Halton sequences

Fig. 1



Figures 2-4 demonstrate some weakness of this sequences that arise in high dimensions.

Fig.2*Fig. 3*

As we will see below the problems are arise with higher dimensions.

Fig. 3 Starting with $n=1$

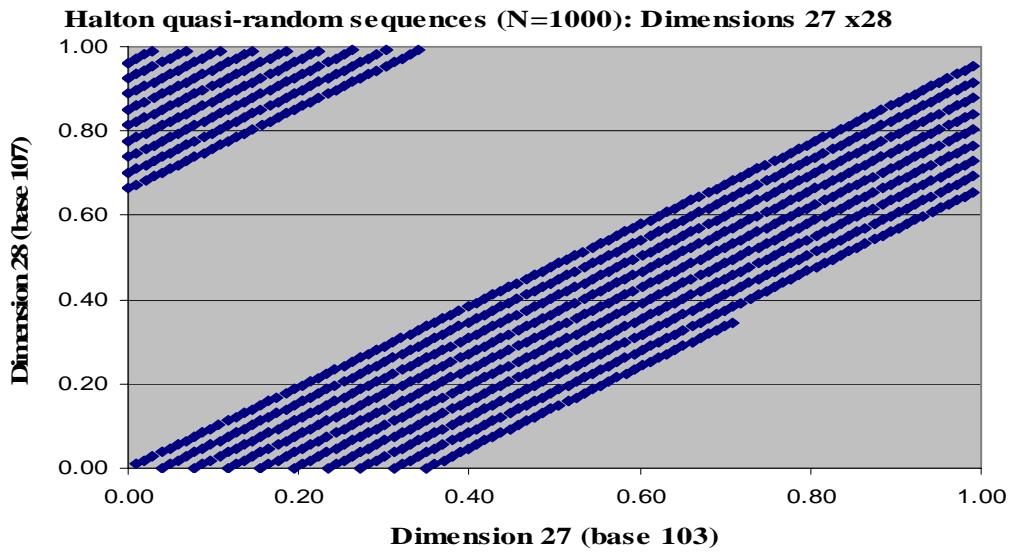
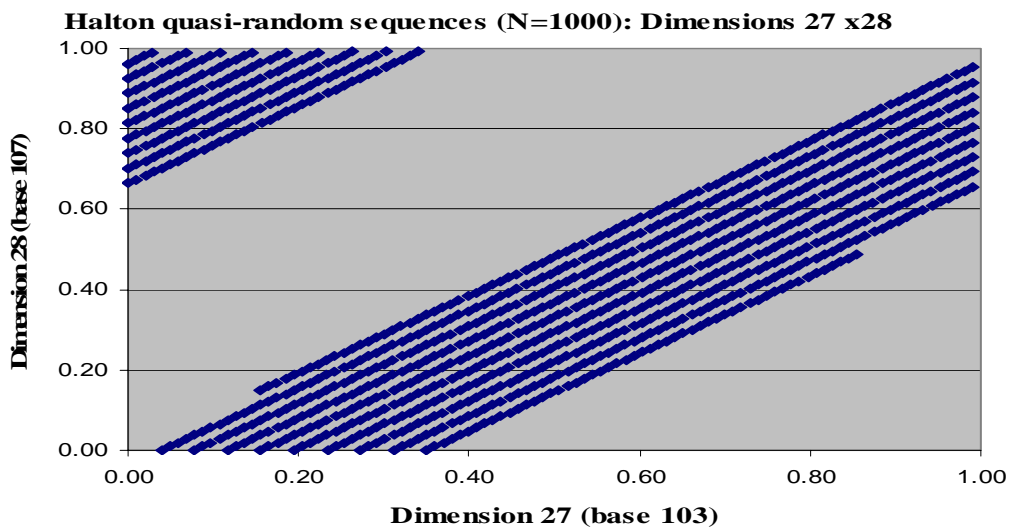


Figure 3 illustrates the problem with Halton sequences: points in successive dimensions are highly correlated and, in high dimensions, the initial points are clustered near zero. All of these problems can lead to poor integral estimates. The second problem can be reduced (see Fig. 4) if take a starting point $n=16$ instead of $n=1$.

Fig. 4 Starting with $n=16$



The sequence preserves its basic properties starting from a different number, so that is not necessary to start from $n = 0$ or $n = 1$. In addition, there are some advantages to cut the first n^* numbers of a sequence to improve the uniformity in higher dimensions, as reported by many authors [[Galanti and Jung](#)].

The major problem for the simple quasi-random sequences is their degradation when the dimension is large. The generation process of uniformly distributed points in $[0,1]^s$ becomes increasingly harder as s increases because the space to fill becomes too large. The high-dimensional Halton sequences exhibit long cycle lengths, due the large prime number base. For example, in the dimension 55, is used the base 257, the 55th prime number. The long cycle length means that the high-dimensional sequence needs a large number of points for an entire walk in the interval $[0, 1)$.

So, Halton sequence becomes unsatisfactory after dimension 14. In practice, due the correlation, many people prefer to avoid the use of Halton sequence for more than 6 or 8 dimensions.

2.4.2 Faure Sequences

The Faure sequence ([Faure], [Joy at al.]) is also a general s -dimensional sequence. Unlike the Halton sequence, all dimensions use the smallest prime p such that $p \geq s$ and $p \geq 2$ as the base. The first dimension of the Faure sequence is the van der Corput sequence in base p . Higher dimensions are permutations of the sequence in the first dimension. For high-dimensional problems the Faure sequence works with van der Corput sequences of long cycle. Long cycles have the problem of higher computational time (compared with shorter cycle sequences). As occurred with high-dimensional Halton sequence, there is the problem of low speed at which the Faure sequence generates increasing finer grid points to cover the unit hypercube. However, this problem is not as severe as in the case of the Halton sequence. For example, if the dimension of the problem is 55, the last Halton sequence (in dimension 55) uses the 55th prime number that is 257, whereas the Faure sequence uses the first prime number after 55, that is a base 59, which is much smaller than 257. So, the "filling in the gaps" in high-dimensions is faster with Faure sequence when compared with the Halton one. General formula for the length of the n -th cycle in base p is $p^n - 1$. For example, for $p = 4$ the first four cycles are 3, 15, 63, and 255.

To construct the Faure s -dimensional sequence we start by representing any integer n in terms of base p as $n = \sum_{i=0}^l a_i^1(n) * p^i$. The first dimension of a Faure point is given by reflecting about the "decimal point", as in the case of van der Corput sequence: $x_n^1 = \Phi_p^1(n) = \sum_{i=0}^l \frac{a_i^1(n)}{p^{i+1}}$.

We will use recursion to find all remaining dimensions (components) of the sequence point. First assume that all $a_i^{k-1}(n)$ are known. Then $a_i^k(n)$ can be obtained from the formula:

$$a_i^k(n) = \sum_{j=i}^l \frac{j!}{i!(j-i)!} a_i^{k-1}(n) \bmod p. \quad (a \bmod b \text{ is the remainder after a number } a \text{ is divided}$$

by divisor b). So, the next level of coefficients is obtained by multiplying $a_i^{k-1}(n)$ by an upper

triangular matrix with elements where $\binom{i}{j} = \frac{j!}{i!(j-i)!}$:

$$\begin{bmatrix} \binom{0}{0} & \binom{0}{1} & \binom{0}{2} & \binom{0}{3} & \dots \\ 0 & \binom{1}{1} & \binom{1}{2} & \binom{1}{3} & \dots \\ 0 & 0 & \binom{2}{2} & \binom{2}{3} & \dots \\ 0 & 0 & 0 & \binom{3}{3} & \dots \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots \\ 0 & 1 & 2 & 3 & \dots \\ 0 & 0 & 1 & 3 & \dots \\ 0 & 0 & 0 & 1 & \dots \end{bmatrix}$$

The successive points in the Faure sequence obtained from

$$x_n^k = \Phi_p^k(n) = \sum_{i=0}^{k-1} \frac{a_i(n)}{p^{i+1}}, \quad 2 \leq k \leq s, n = \overline{1, N}.$$

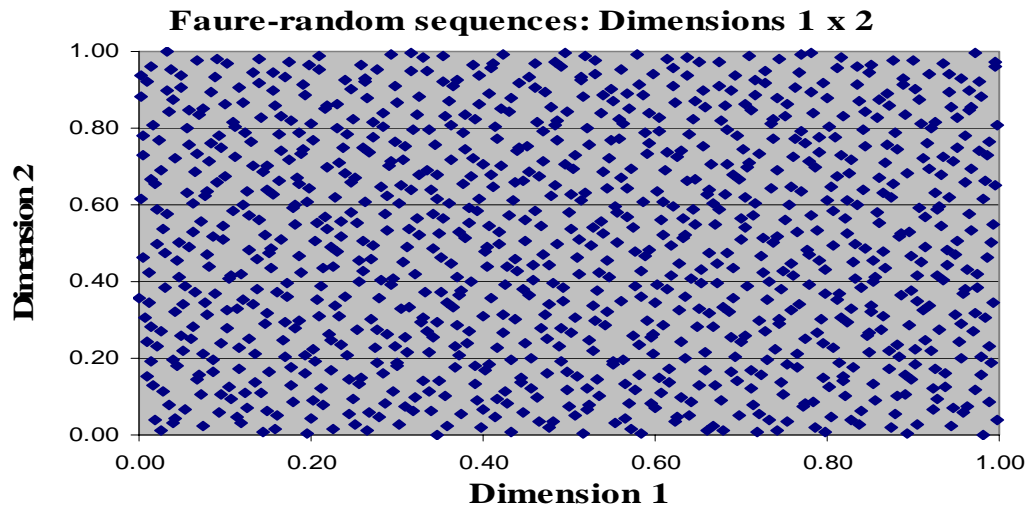
This recursive procedure permits us to generate s components (dimensions) corresponding to each Faure point n ($n = \overline{1, N}$, N is a number of simulations) in the Faure sequence with p as base ($p \geq s$).

Table 2. Faure sequences for first 3 dimensions

	<i>Dim=1</i> (Base 3)	<i>Dim=2</i> (Base 3)	<i>Dim=3</i> (Base 3)	<i>Dim=1</i> (Base 3)	<i>Dim=2</i> (Base 3)	<i>Dim=3</i> (Base 3)
	$a_0(n)$	$a_1(n)$	$a_2(n)$	x_n^1	x_n^2	x_n^3
$n=1$	1	0	0	1/3	1/3	1/3
$n=2$	2	0	0	2/3	2/3	2/3
$n=3$	0	1	0	1/9	4/9	7/9
$n=4$	1	1	0	4/9	7/9	1/9
$n=5$	2	1	0	7/9	1/9	4/9
$n=6$	0	2	0	2/9	8/9	5/9
$n=7$	1	2	0	5/9	2/9	8/9
$n=8$	2	0	1	8/9	5/9	2/9

Figures below represent Faure sequences for different dimensions.

Fig. 5 Faure sequences, base 3, $N=1000$



The problems with Faure sequences arise with the dimension. Higher dimension requires a larger prime number taken as base and causes higher correlation between two neighboring components of the points.

Fig.6

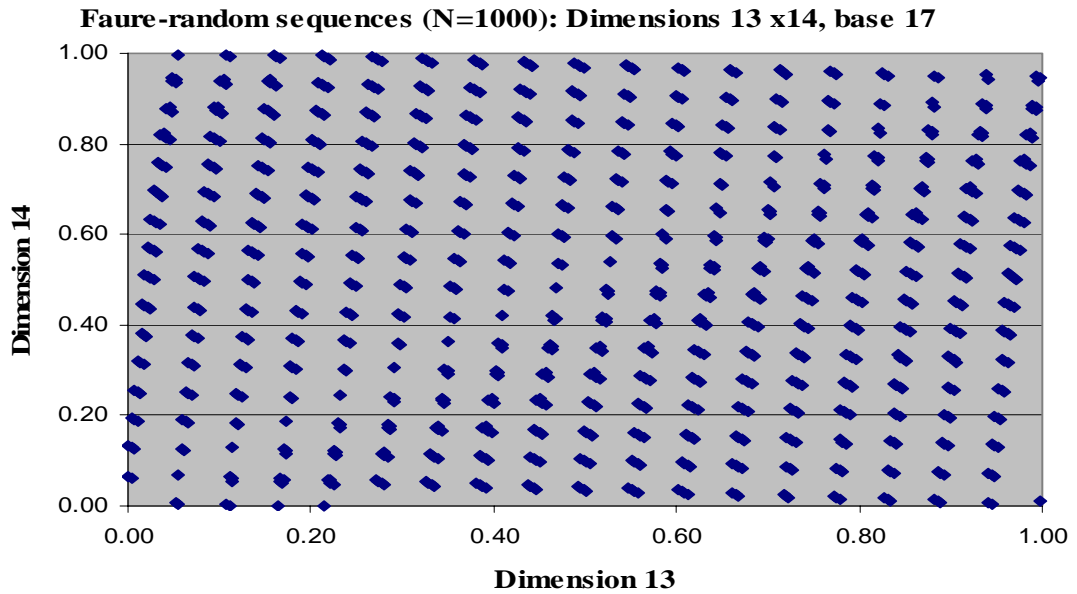
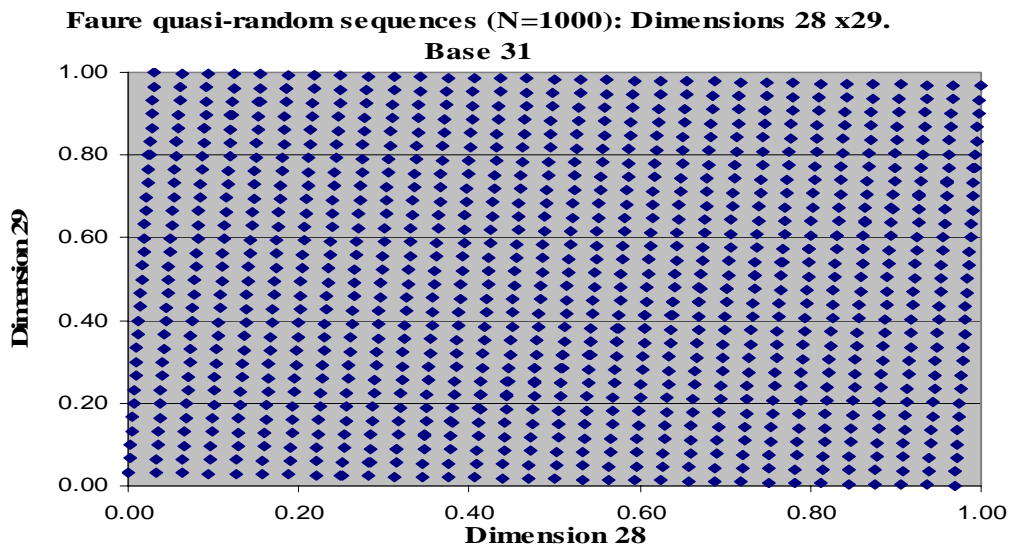


Fig.7



Faure sequence is build the way that with each additional dimension number of rows that are absolutely the same is growing. As reported in [[Galanti and Jung](#)], among others, the Faure sequence suffers from the *problem of start-up*. In particular, for high-dimension and low starting values n_0 , the Faure numbers exhibits clustering about zero. In order to reduce this problem, Faure suggests discarding the first $n = (p^4 - 1)$ points, where p is the base. The start-up problem is reported also for other sequences with the same practical suggestion of discarding some initial numbers from the sequence. The Faure sequence exhibits high-dimensional degradation at approximately the 25th dimension. But according our experience, what is surprising that at dimension 30 these sequences behave better than at dimension 14. By eliminating the first n points we can get “empty spots” in our sequences that cause decreasing of uniformness.

There are some variations for this class of sequences. [Tezuka](#) formulated the *Generalized Faure Sequence*, based on Halton sequence but using polynomials for reordering. According to [Traub](#), generalized Faure method converges significantly faster than Monte Carlo and simple Faure method.

2.4.3 Sobol Sequences

In the general s -dimensional Sobol' sequence [[Sobol](#)] for all dimensions uses the prime number 2 as the base. The first dimension of the Sobol' sequence is the van der Corput sequence in base 2, and higher dimensions are permutations of the sequence of the first dimension. Permutations depend on a set of direction numbers v_i . The numbers $v_i = \frac{m_i}{2^i}$, $i = \overline{1, w}$, form a sequence of binary fractions with w bits after the binary point, where $0 < m_i < 2^i$ are odd integers. The Sobol' sequence is not uniquely defined until all of these direction numbers are defined.

In the Sobol algorithm [[Sobol](#)] a one-dimensional Sobol' sequence is generated by $x_n = a_1 v_1 \oplus a_2 v_2 \oplus \dots \oplus a_w v_w$, $n \geq 0$, where $n = \sum_{i=0}^{\lfloor \log n \rfloor} a_i 2^i$ is the binary representation of n and \oplus denotes a bit-by-bit exclusive-or-operation (XOR). For example, $n=5$ is 101 in base 2. If $v_1 = 0.1$, $v_2 = 0.11$, $v_3 = 0.111$, then $x_5 = 1 * 0.1 \oplus 0 * 0.11 \oplus 1 * 0.111 = 0.1 \oplus 0 \oplus 0.111 = 0.011$.

As n increments different "ones" of the v_i 's flash in and out of x_n on different time scales. v_1 alternates between being present and absent most quickly, while v_n goes from present to absent (or vice versa) only every 2^{n-1} steps.

For the construction of Sobol sequences the notion of Grey codes are used.

The *Gray Code* of an integer j is defined as $G(j) = j \oplus \text{int} \left[\frac{j}{2} \right]$, where $\text{int}[x]$ represents the largest integer inferior or equal to x . Notice that $G(j-1)$ and $G(j)$ differ, in their binary representations, only in the digit relative to the rightmost zero in the binary representation of $(j-1)$. Therefore, in the construction of the j^{th} Sobol number, the induction is the same as XORing all the direction numbers associated to the unit bits of $G(j)$. For example,

$$G(2) = 2 \oplus 1 = (1 * 2^1 + 0) \oplus 1 = 10 \oplus 1 = 11 = 3).$$

To construct of the Sobol sequence for each dimension k we should follow as

1. Choose an integer x randomly, for instance, $x = 2$. This number defines the starting point of the sequence (the "seed" of the process).
2. Compute the Gray Code of x , $G(x)$, as explained above.
3. Transform $G(x)$ to its binary representation ($G(2) = 3 = 1 * 2^1 + 1 * 2^0 = 11$).
4. Sum bit by bit (XOR) the direction numbers associated with the digits of $G(x)$ (in binary representation) that are different from zero. In the example, counting from right to left, the bits of $G(x)$ different from zero are the first and second. Therefore XOR has to be done with the first (0.1) and the second (0.11) direction numbers:

$$y(2; k) = 0.10 \oplus 0.11 = 0.01$$

5. Transform the resulting number into a decimal number contained in $(0; 1)$, by multiplying each digit of that binary representation by 2^{-l} , where l is the position of the digit in the decimal part of number, counting from left to right. Add up the terms. The result is the first Sobol number in dimension k : $s(2, k) = 0 \times 2^{-1} + 1 \times 2^{-2} = 0.250$.

[Antonov and Saleev](#) proposed a shifting of the original Sobol' sequence, which can be generated much faster while preserving good convergence properties. They show that instead of using the bits of the integer n to select direction numbers, the bits of the *Gray code* of n could be used. In their algorithm, the sequence of direction numbers v_i is generated by a primitive polynomial with coefficients in the field Z_2 with elements $\{0, 1\}$. In case of the primitive polynomial $P(x) = x^n + b_1x^{n-1} + \dots + b_{n-1}x + 1$ of degree n in $Z_2[x]$, the direction numbers can be obtained from the recurrent formula:

$$v_i = b_1v_{i-1} \oplus b_2v_{i-2} \oplus \dots \oplus b_{n-1}v_{i-n+1} \oplus v_{i-n} \oplus (v_{i-n} / 2^n), i > n,$$

where the last term v_{i-n} is shifted right n places.

Instead of XORing several directions numbers, the j -th Sobol can be obtained from the $(j-1)$ -th using just one direction number: $y(j; k)_2 = y(j-1; \kappa)_2 \oplus v_{jk}$, where v_{jk} is the direction number associated with the rightmost zero in the binary representation of $(j-1)$. In the example, $210 = 102$, therefore the rightmost zero is encountered in the first position and one should XOR the first direction number of the k -th dimension with $y(2; k)$. The second number in the sequence will be:

$$y(3; k) = 0.01 \oplus 0.10 = 0.11$$

$$s(3; k) = 1 \times 2^{-1} + 1 \times 2^{-2} = 0.750$$

Similarly, $310 = 112$ and it is necessary to add a leading zero to put our hands on the rightmost zero in the binary representation of 3, that is, the 10 digit in the third position. Therefore we need to XOR the third direction number. The third number in the sequence is:

$$y(4; k) = 0.110 \oplus 0.101 = 0.011$$

$$s(4; k) = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 0.375$$

The fourth number depends on $y(4; k)$ and on the first direction number, because the rightmost zero in the binary representation of 4 is in the first position: $410 = 1002$

$$y(5; k)_2 = 0.011 \oplus 0.100 = 0.111$$

$$s(5; k) = 1 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 = 0.875$$

And so forth... The numbers "fill" the gaps in the interval $(0; 1)$ looking for empty spaces, as if the procedure knew where the positions of all prior numbers were.

To construct a multi-dimensional Sobol' sequence, consider P_1, P_2, \dots, P_s to be s primitive polynomials in $Z_2[x]$. Denote [\[Paskov\]](#) $\{x_n^i\}_{n=1}^{\infty}$ the sequence of one-dimensional Sobol' points generated by the polynomial P_i . Then the sequence of s -dimensional Sobol's points is defined as $x_n = (x_n^1, x_n^2, \dots, x_n^s)$.

Table 3. Sobol three-dimension sequences (Start No=0)

	<i>Dim=1</i> (Base 2)	<i>Dim=2</i> (Base 2)	<i>Dim=3</i> (Base 2)
<i>n=1</i>	1/2	1/2	1/2
<i>n=2</i>	3/4	1/4	3/4
<i>n=3</i>	1/4	3/4	1/4
<i>n=4</i>	3/8	3/8	5/8
<i>n=5</i>	7/8	7/8	1/8
<i>n=6</i>	5/8	1/8	3/8
<i>n=7</i>	1/8	5/8	7/8
<i>n=8</i>	3/16	5/16	5/16

Figures below demonstrate how quality of Sobol sequences varies in different dimensions.

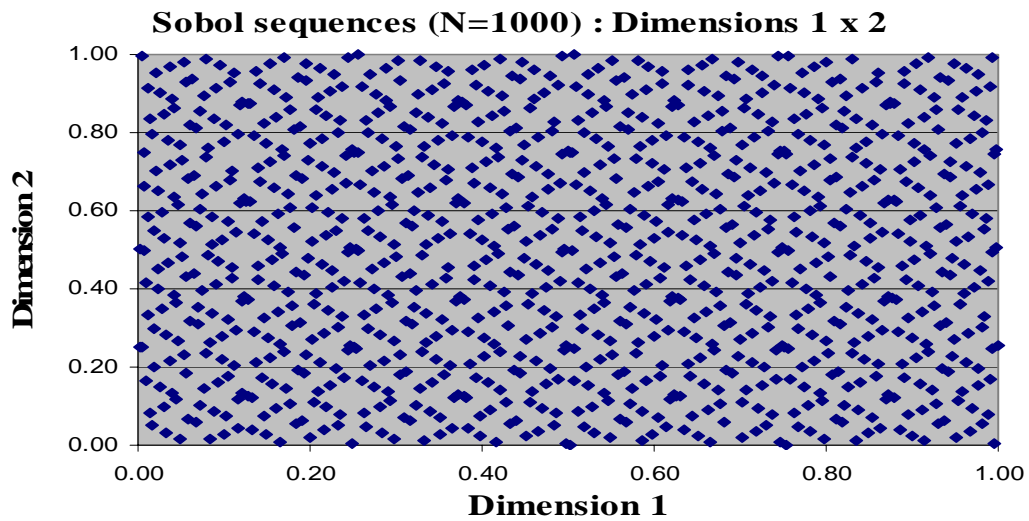
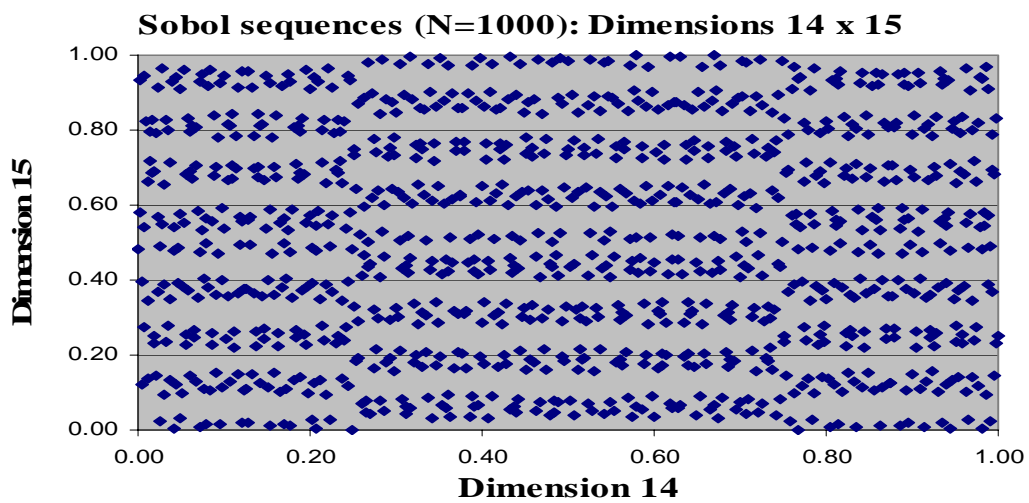
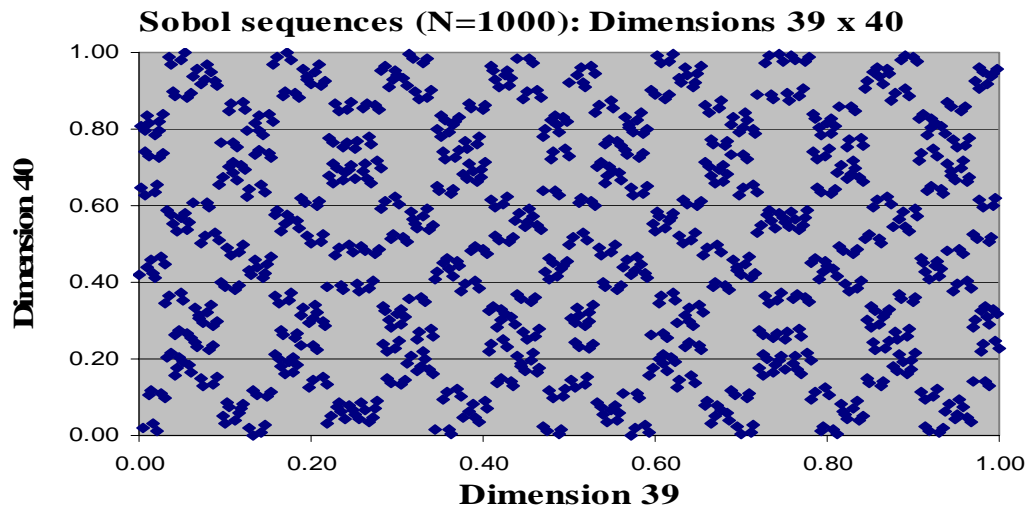
Fig.8**Fig.9**

Fig.10



Sobol sequences possess a good uniformness and require a little time to compute them. [Galanti and Jung](#) showed that the Sobol sequences presented no degradation at all up to the dimension 260.

Each dimension k of the Sobol sequence is created with the use of a different primitive polynomial. There are several tables of these polynomials available in the literature. [Press et al.](#) list 150 primitive polynomials, allowing the construction of Sobol sequences up to 150 dimensions. Jaeckel [2002] offers a CD containing more than 8 million polynomials. These should suffice for nearly all practical problems in finance.

Alternative Heuristic Random Method of Generation of the Initial Direction Numbers:

1. For each dimension k , using a pseudo-random generator, draw d_k uniform numbers u_{ik} such that $m_{ik} = \max(\text{int}[u_{ik} \times 2^i - 1]; 1)$ for $0 < i < d_k$ is odd (simply keep drawing u_{ik} until the condition is met), and use $v_{ik} = \frac{m_{ik}}{2^i}$ as the direction number for dimension k .

2. Generate randomly a seed for each dimension. It is preferable to choose large numbers as seeds; this will force the algorithm to use direction numbers of higher orders in the w -bit long array for each dimension (this enhances the power of randomization of the algorithm).

This method was used by [Silva](#) to generate 2500-dimensional Sobol sequences with 2047 points each. The uniform numbers in consecutive dimensions covered the unit square very well for nearly all the 2500 dimensions. The number of dimensions that can be tackled with this method depends only on the amount of available primitive polynomials, currently around 8 million. Thus it seems that the curse of dimensionality could be broken in practice.

2.4.4 A Hybrid Quasi-Monte Carlo Approach

The following Hybrid Quasi-Monte Carlo approach is very simple and it is inspired by the Latin-hypercube technique named *stratified sampling without replacement* [Vose, 2000]. For a problem with s dimensions and N simulations:

1. Generate a vector-column of N random or quasi-random numbers as the basic vector. For example, we can use the van der Corput sequence in base 2 for the first dimension.
2. For each dimension $k, k = \overline{1, s}$, use this vector but with *independent random permutations* for the elements of this vector.

Table 4. Hybrid Quasi three-dimension sequences

	<i>Dim=1</i> <i>(Base 2)</i>	<i>Dim=2</i> <i>(Base 2)</i>	<i>Dim=3</i> <i>(Base 2)</i>
<i>n=1</i>	1/2	1/2	1/4
<i>n=2</i>	1/4	1/8	3/8
<i>n=3</i>	3/4	3/4	1/2
<i>n=4</i>	1/8	5/8	1/16
<i>n=5</i>	5/8	3/8	5/8
<i>n=6</i>	3/8	1/4	7/8
<i>n=7</i>	7/8	1/16	3/4
<i>n=8</i>	1/16	7/8	1/8

As we can notice from the table each column consists of the same elements but in different random order. It uses the same base 2 for all dimensions as Sobol sequences, but permutations of first columns are random.

Fig. 11

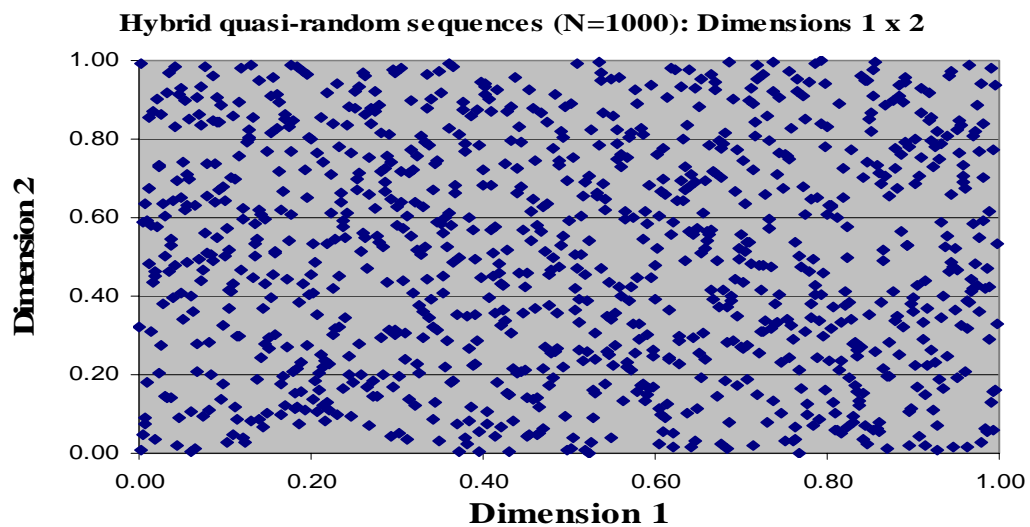


Fig.12

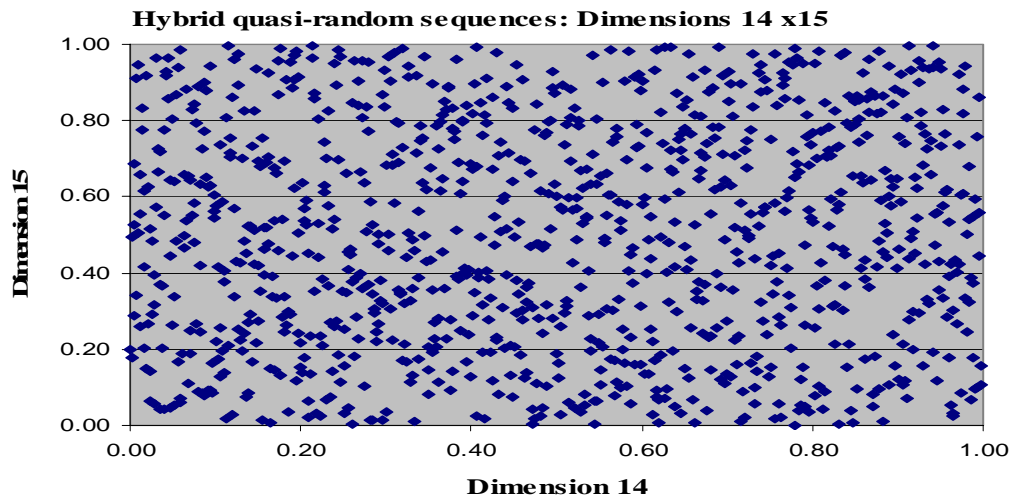
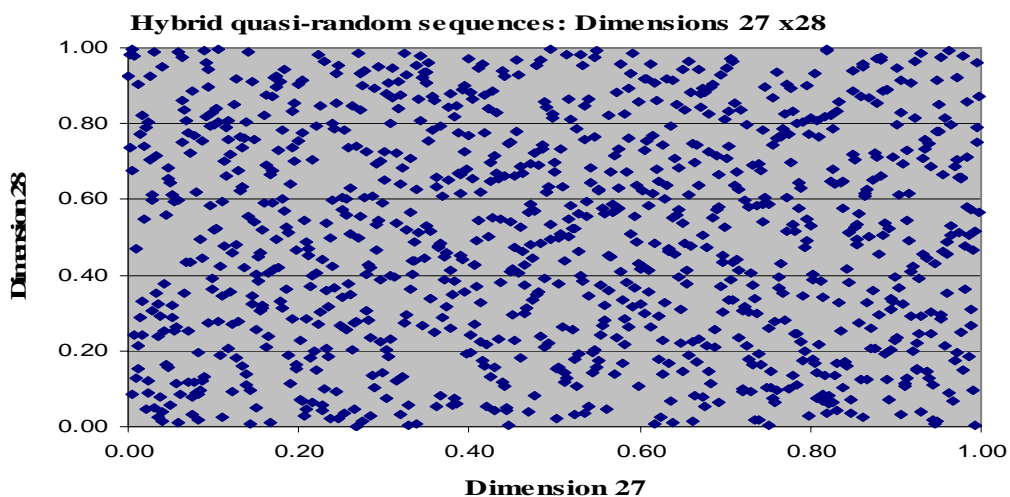


Fig. 13



Because the Hybrid quasi-points built as permutations of Corput sequence with base 2, this method does not experience problems with autocorrelation in high-dimension environment.

The random permutation idea is to break the correlations (cyclicity) of the sequence of s low-discrepancy vectors of N components each. The resulting vector after the permutation preserves its low discrepancy property of evenly distributed numbers in the interval $[0, 1)$, because the numbers are the same in another order, and in addition the sequence in the same row from the matrix (that is, in the same sample path) gains the property of independence or at least become closer for practical purposes. From the graphs above we can notice that the degradation of low-discrepancy sequence in high-dimensional case is not occurring for hybrid quasi-random sequence. In terms of high-dimensional clustering this means that the hybrid quasi-random sequence is not worse than the pseudo-random sequence in multi-dimensional case. The multi-dimensional sequence being more evenly dispersed in each individual one dimension has higher potential to be evenly dispersed at higher dimension if it is possible to eliminate the correlation between dimensions. This job is performed by the random permutation, which makes a random association

between quasi-random numbers (van der Corput with base 2, for example) for every 2D projection. By using these hybrid QR sequences we have the guarantee of nice cross-temporal statistical properties: for any time instant t we have a van der Corput base 2 (permuted) sequence, which we see before is a much better approximation for $U[0, 1]$ than the pseudo-random sequence, without the problem of degradation observed in the standard (non-hybrid) quasi-random sequences for higher dimensions.

2.4.5 (t, m, s) -Nets and (t, s) -Sequences

The quasi-random points that we discussed in this chapter are special cases of (t, m, s) -nets and (t, s) -sequences [Niederreiter].

An elementary interval in base b is a set of the form

$$J = \prod_{j=1}^s \left[\frac{a_j}{b^{k_j}}, \frac{a_j + 1}{b^{k_j}} \right) \subseteq I^s$$

for non-negative integers a_j, k_j with $a_j < b^{k_j}$. For $t \geq 0$, a sequence $N = b^m$ points x_n is a (t, m, s) -net in base b if every elementary interval J in base b of volume b^{t-m} has

$$R_N(J) = \frac{1}{N} \sum_{n=1}^N X_J(x_n) - m(J) = 0.$$

(X_J is a characteristic function of the set J , and $m(J)$ is its volume).

The smaller t is the better, given m, s , and b . Also, a smaller base is preferable, since it implies that uniformity holds over shorter subsequences.

An infinitive sequence $(x_n)_{n=1}^{\infty}$ is a (t, s) -sequence in base b if each finite sequence $(x_n)_{n=Ab^n, j}^{(A-1)b^n}$ with $A \geq 0$ is a (t, m, s) -net in base b for all $m \geq 0$.

Other words, a (t, m, s) -net in base b (with $0 \leq t \leq m$) is a set of b^m points in the s -dimensional hypercube such that every elementary interval of volume b^{t-m} contains b^t points.

The net property becomes relevant for $m > t$, that is $N \geq b^{t+1}$. Below this value of N , any sequence of points in I^s , even identical points, is consistent with the net property [Caflisch]. The smallest N at which the net property constrains some fully s -dimensional elementary sub-interval (one with all $k_j > 0$) is b^{t+s} . The asymptotic rate for the discrepancy of nets should therefore start at around $N = b^{t+s}$.

Faure points are $(0, s)$ -sequences in prime bases not less than s . Sobol points are (t, s) -sequences in base 2. For $s=360$ the value of t is quite large for Sobol' sequences (according to Niederreiter in the thousands). For $s=360$ the smallest value of b for Faure sequence is 361. Therefore, Faure points achieve the smallest value of t , but at the expense of a large base.

2.4.5 Comparison Low Discrepancy Sequences

To compare low-discrepancy sequences, several criteria were used.

1. *Uniformity*. We compute methods on the basis of how uniformly they fill the hypercube $[0,1)^s$. Better uniformity leads to smaller integration error at the same sample size.

The following table illustrates the nice statistical properties of quasi-random sequence (used a *van der Corput sequence in base p* compared with the (theoretical) Uniform $[0, 1]$ distribution, and pseudo-random sequence (generate with Excel).

Table 5. Statistical Properties of Low Discrepancy points (LDP) vs. Linear Congruent Generator (LCG)

Comparison (n=1000) for Uniform Distribution U[0,1]							
	Theoretical	LCG	LDP				
		Pseudo-Random	Corput Base 2	Halton Base 3	Faure Base 3	Sobol Dim 2	Hybrid Dim 2
Mean	0.50000	0.49822	0.49951	0.49899	0.49928	0.50040	0.49951
Variance	0.08333	0.08573	0.08339	0.08332	0.08314	0.08318	0.08339
Skew	0.00000	-0.00936	0.00000	-0.00112	0.00069	-0.00051	0.00000
Kurtosis	-1.20000	-1.24690	-1.20087	-1.19902	-1.19871	-1.19799	-1.20087
Max	1.00000	0.99845	0.99805	0.99863	0.99863	0.99902	0.99805
Min	0.00000	0.00195	0.00098	0.00046	0.00046	0.00098	0.00098

As we can notice from this table, the LDP sets routinely outperformed the LCG set. Sobol sequence has better uniform properties regarding to the mean and maximum number comparing to other LDS, whereas Faure sequence has the lowest variance at base 3. Corput sequences with base 2 and therefore Hybrid quasi sequences perform better regarding to skew and kurtosis.

Hence, quasi-random sequence presents a better performance than typical pseudo-random sequences for all four probabilistic moments, indicating that quasi-random sequence is more representative of $U[0, 1]$ than pseudo-random numbers.

2. *Speed of generation*. We compare methods on the basic of the computer time it takes to generate a given number of s -dimensional points.

Table 6. Speed of generation of different sequences (N is a number of M-dimensional points, where M is the dimension). All numbers in this table are in seconds.

Sequence	N=1,000,000; M=6	N=1,000,000; M=12	Times of time increasing
<i>Pseudo-Random</i>	3	6	2.5
<i>Halton</i>	8	14	1.75
<i>Faure</i>	79	200	2.53
<i>Sobol</i>	2	4	2
<i>Hybrid</i>	10	17	1.7

As we can notice from this table, the Sobol sequence can be generated significantly faster than the Faure sequence and faster than most pseudo-random number methods. Advantages the Sobol sequence over Faure and Halton sequences is due to the use base 2 for all dimensions. So, there is some computational time advantage due the shorter cycle length. The time to construct Hybrid sequences is larger than for Halton sequences due to additional operations needed to calculate uniform permutations of the one-dimension sequence (base 2). Faure sequences consume much more times that all other tested here sequences. According to [Traub](#), the generalized Faure method always converges as fast as the modified Sobol method and often faster, but this method is beyond our work.

3. *Standard error.* We compare methods on the basis of the standard error obtained by implementation these methods to practice.

As an example, we compute the standard error resulting in implementing these methods to European call evaluation.

Table 7. Standard error of different sequences for different number of paths (N) and dimension M (M=6) in European call evaluation.

Sequence	N=1,000; M=6	N=10,000; M=6	Times of SE decreasing
<i>Pseudo-Random</i>	0.0394	0.0192	2.05
<i>Halton</i>	0.0333	0.0108	3.08
<i>Faure</i>	0.0241	0.0085	2.84
<i>Sobol</i>	0.0058	0.0046	1.26
<i>Hybrid</i>	0.0381	0.0236	1.6

As we can notice from this table, the Sobol sequence gives the least standard error, comparing to other methods. All quasi-random sequences perform better than pseudo-random, but Hybrid method gives the worst error among other low-discrepancy sequences. Faure sequences are outperforms all other sequences but Sobol ones, but we should remember from previous comments that Faure sequences consume more time.

Sobol sequence can achieve the 1% accuracy (for the above example) with N=700 samples while pseudorandom sampling requires nearly 15,000 samples and consume 8 times more time than Sobol sequences to get the same accuracy.

Actually, to evaluate a European call option it is enough to take just one dimension (in this case all low-discrepancy sequences will lead to the same result because they all use de van Corput sequence with base 2 for this dimension. The table 8 demonstrates power of QMC methods comparing simple MC in this case.

Table 8. Standard error of different number of paths (N) and dimension M (M=1) in European call evaluation.

Sequence	N=1,000; M=1	N=10,000; M=1	N=25,000; M=1
<i>MC</i>	0.0076	0.0132	0.0131
<i>QMC</i>	0.0111	0.0044	0.0039

As we can notice from the table to improve accuracy we would require more pseudo-random numbers than low-discrepancy ones.

4. *High dimensional clustering.* We compare methods on the basis of high dimensional clustering near zero. If the clustering occurs for some sequence it would result in decreasing its effectiveness and reliability.

As was noted and demonstrated in chapters 2.4.1-2.4.4 the major problem for the simple quasi-random sequences is their degradation when the dimension is large. The generation process of uniformly distributed points in $[0,1]^s$ becomes increasingly harder as s increases because the space to fill becomes too large. While Sobol sequence "fills in the gaps" at higher rate in high-dimensional problems (due its shorter cycle), there is the problem that at high-dimensions the Sobol points tend to repeat themselves across dimensions, resulting in high-dimensional clustering. The traditional solution is to discard the first n points. As it was observed from the experience by the author the Faure sequences perform better if first $n=Prime\ number\ used\ to\ generate\ this\ sequence$ are discard. Boyle, Broadie, and Glasserman [[Boyle et al.](#)] discard the first 64 points for Sobol sequence, but this number can be chosen arbitrary. The Sobol sequence appears to resist more to the high-dimensional degradation. [Galanti and Jung](#) showed that the Sobol sequences presented no degradation at all up to the dimension 260. According our analysis Halton and the Faure sequences suffer from this problem with much lower dimension. Hybrid sequences as well as pseudo-random sequences do not have such problem, but as we noted before, they result in lower standard error and could not be considered as a good alternative to low-discrepancy sequences.

5. *Correlation between nearly dimensions.* We compare methods on the basis of correlation between neighboring dimensions. If the neighboring dimensions are highly correlated it will lead to decrease its quality and multi-dimensional uniformity.

As was noted and demonstrated in chapters 2.4.1-2.4.4 the nearly dimensions of some quasi-random sequences in high-dimensional problems are highly correlated. Fig. 3 demonstrates such problem with Halton sequences. So, Halton sequence becomes unsatisfactory after dimension 14. In practice, due the correlation, many people prefer to avoid the use of Halton sequence for more than 6 or 8 dimensions. Because the Hybrid quasi-points built as permutations of Corput sequence with base 2, this method does not experience problems with autocorrelation in high-dimension environment. Fig. 6-7 demonstrate the problem with Faure sequences: the Faure sequence exhibits high-dimensional degradation at approximately the 25th dimension. According to Fig. 9-10 Sobol sequences also express some correlation between neighboring dimensions, but this correlation is lower than for other low-discrepancy sequences and started at higher dimension.

As the result of different comparisons that are discussed in this chapter some summary and recommendations could be made.

The Halton sequence is dominated by the Faure and Sobol sequences. Although an implementation of Halton method is much easier than Sobol and Faure methods, it is not wise to use them in high dimensions. As will be shown later in finance application its performance falls dramatically with increasing dimensions. Hybrid Quasi method shows no degradation in high dimensions, but its uniform properties are worse than of other LD sequences. For low dimensions, all LD sequences can be successfully used. But with higher dimensions, it might be good to use Sobol sequences or the generalized Faure method to be positive in fast convergence and reliability.

2.5 Generating of Normally Distributed Sequences

The uniform distribution can be generated with either pseudo random number or quasi-random numbers. Algorithms are available to transform a uniform distribution in any other distribution. The main and direct way to do this transformation is by the *cumulative distribution function inversion*. The most important distribution for financial and other applications is the Standard Normal Distribution.

For standard normal distribution, the cumulative distribution is:

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

In a simulation problem, we have the $Y(x)$ from uniform distribution, $Y(x) \sim U[0, 1]$, but we want the corresponding x samples for any distribution, e.g., the normal distribution. In other words, we want $x(Y)$, the inverse of the cumulative distribution. Hence, given one uniformly distributed sample u we want to obtain the value $x(u)$, that is, the correspondent sample for the desirable probability distribution.

The best-known general way of generating normally distributed points is by using the *Box-Muller algorithm* (see [Press at al.]). However, for low discrepancy sequences has been reported that this is not a good way to go because it damage the low discrepancy properties of the sequence (alters the order of the sequence or scrambles the sequence uniformity) [Moro], [Galanti and Jung]. In addition, it has been reported that the Box-Muller algorithm is slower than the *Moro's inversion* described below. The traditional normal *inversion* algorithm is given by [Beasley and Springer] (see [Moro]). However, this algorithm is not very good for the tails of the normal distribution. The best way to obtain the inversion from $U[0, 1]$ to normal distribution is by using an algorithm presented in a famous short paper of [Moro]. Moro presented a hybrid algorithm: he uses the Beasley & Springer algorithm [Beasley and Springer] for the central part of the normal distribution and another algorithm for the *tails of the distribution*. Moro modeled the distribution tails using truncated *Chebyshev series*. The Moro's algorithm divides the domain for U (where U = uniform sample) into two regions:

1. The central region of the distribution, $|U| \leq 0.42$, $U = x - 0.5$, is modeled as in

$$[\text{Beasley and Springer}] : \Phi^{-1}(x) = U \frac{\sum_{n=0}^3 a_n U^{2n}}{\sum_{n=0}^4 b_n U^{2n}} \quad \text{where } a_n, b_n \text{ are from the table below.}$$

n	a_n	b_n
0	2.50662823884	1.00
1	-18.61500062529	-8.4735109309
2	41.39119773534	23.08336743743
3	-25.44106049637	-21.06224101826
4		3.13082909833

2. The tails of the distribution, $|U| > 0.42$, are modeled by a truncated Chebyshev series as

$$\Phi^{-1}(x) = \begin{cases} \sum_{n=0}^8 c_n T_n(z) - \frac{c_0}{2}, & \text{if } U > 0 \\ \frac{c_0}{2} - \sum_{n=0}^8 c_n T_n(z), & \text{if } U \leq 0 \end{cases}; z = k_1 \cdot [2 \cdot \ln(-\ln(0.5 - |U|)) - k_2].$$

The constants k_1 and k_2 are chosen such that $z = -1$ when $\Phi(x) = 0.92$ and $z = 1$ when $\Phi(x) = 1 - 10^{-12}$.

n	c_n	k_n
0	7.7108870705487895	
1	2.7772013533685169	0.4179886424926431
2	0.3614964129261002	4.2454686881376569
3	0.0373418233434554	
4	0.0028297143036967	
5	0.0001625716917922	
6	0.0000080173304740	
7	0.0000003840919865	
8	9.9999999129707170	

That procedure has high accuracy for all values of x in the interval $[10^{-10}, 1 - 10^{-10}]$ while the speed remains as with a simple rational approximation.

To compare Moro's Inversion with Standard Excel Inversion (STANDSINV()), we use a van der Corput quasi-random sequence in base 2 and random sequence. The table 9 shows that both inversions (Moro and Excel) in case of Quasi-Random sequences presented practically the same (acceptable) accuracy, but not for pseudo-random where Moro's Inversion is more accurate for all parameters but variance with standard deviation.

Table 9. Statistical Properties of Quasi-Random Inversion to Standard NORMSDIST ($N=1000$)

	Normal [0,1]	Moro's Inversion		Excel's Inversion	
		Pseudo-Random	Quasi-Random	Pseudo-Random	Quasi-Random
Mean	0	-0.0111	0.0002	-0.0235	0.0003
Median	0	0.0009	0.0000	0.0033	0.0000
St Deviation	1	0.9779	0.9886	1.0085	0.9940
Variance	1	0.9562	0.9772	1.0170	0.9880
Skewness	0	0.0533	0.0097	-0.1318	0.0121
Kurtosis	3	2.8451	2.7870	3.1571	2.8914

The Excel inversion weak side is in the tails accuracy. To prove that we can take uniform numbers (\mathbf{u}) successively closer to zero so that the inverse numbers are successively more negative (in the

left tail of the Normal). The "exact" numbers and the idea for this kind of test is drawn from the paper of McCullough & Wilson (2001, p.6).

Table 10. Comparison of Normal Standard Inversion Functions

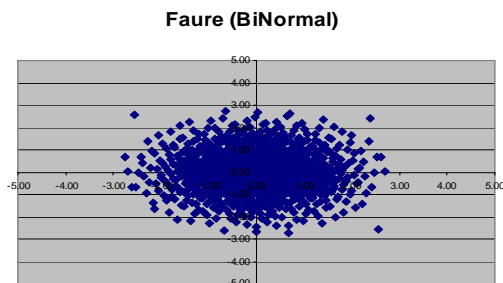
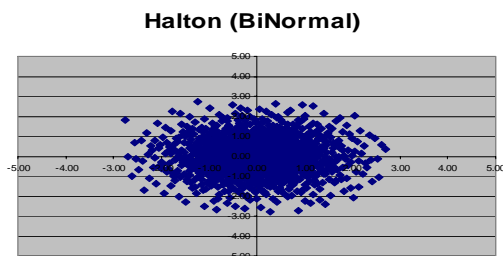
u	Normal [0,1]	Moro's Inversion	Excel NORMSINV
0.001	-3.09023	-3.09023	-3.09025
0.0001	-3.71902	-3.71902	-3.71909
0.00001	-4.26489	-4.26489	-4.26504
0.000001	-4.75342	-4.75342	-4.75367
0.0000003	-4.99122	-4.99122	-4.99152

On the contrary, note that the Moro's inversion get the same result of the column "exact" for the precision of 5 decimal digits. This nice performance of the Moro's inversion for the tails can make a significant difference for real options valuation. The "tail performance" is important for a large number of simulations (so that non-zero low discrepancy numbers get closer to the limits of the interval).

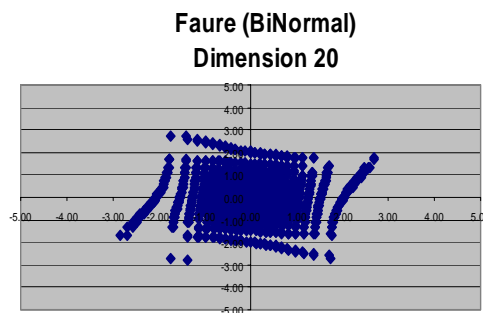
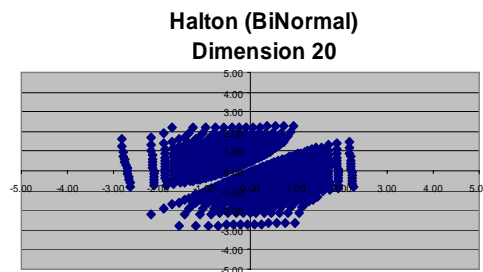
To successfully use LD sequences in finance, transformed numbers from uniform distribution to normal must posses a good normality. The graphs below demonstrate transformed standard normal samples using Moro's method.

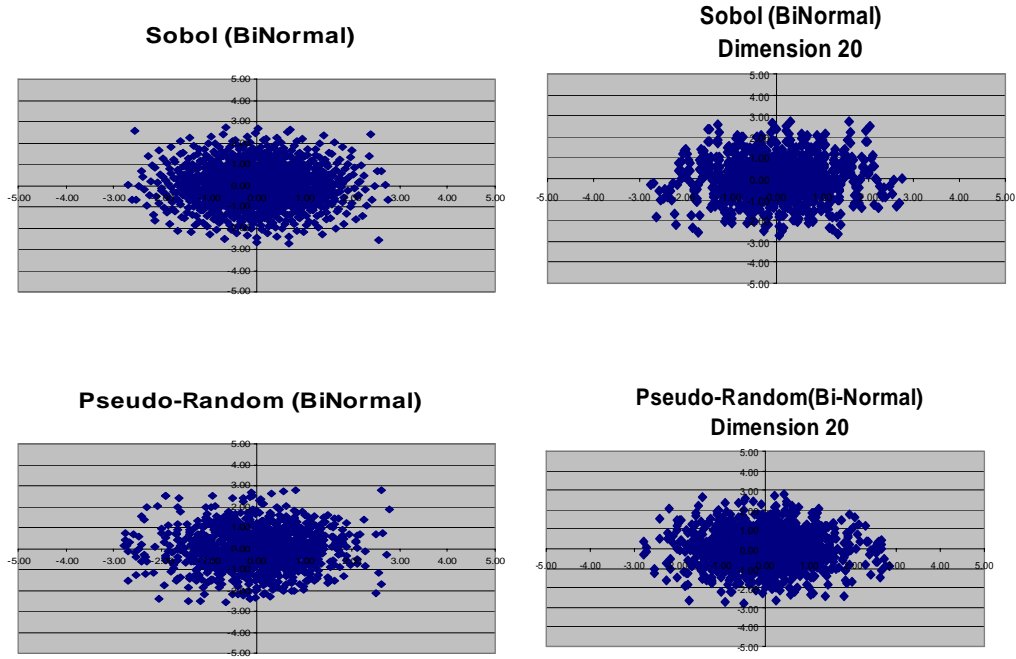
Fig. 15 Transformed standard normal samples using Moro's method in different dimensions.

Dimension 2:



Dimension 20:





The graphs above confirm our claims about the usefulness of Sobol sequences to get a good practical implementation of normality.

CHAPTER 3

GENERATION OF MATRICES OF BROWNIAN MOTION PATHS

The generation of paths usually takes several steps:

1. The generation of a (N by s) - matrix of (pseudo-) random numbers, where N is the number of samples and s is the dimension. Standard Monte Carlo or Quasi-Monte Carlo (Sobol, Faure, Halton etc. sequences) methods can be used for this purpose.
2. Transformation of the matrix on the first step to a matrix of non-uniform random variables. To obtain normal random numbers from uniform numbers Moro's inversion algorithm or Box-Muller transformation could be used. Once we know how to generate a sample Z from the standard normal distribution, we can generate a sample X from the normal distribution with mean μ and standard deviation σ by setting $X = \mu + \sigma \cdot Z$.
3. The use of the matrix of the non-uniform random to build paths.

For example, in more basic option problem we can think the number of dimensions s as the number of discrete time intervals of one sample path, so $s = T/\Delta t$, where T is typically the expiration of the options or the horizon of interest. The number of iterations N is the number of sample paths. The number of dimensions is at least the number of time-steps (that is $T/\Delta t$). So, across the paths at one specific time instant, we want a good *uniform sample numbers* in order to generate for example a good Normal distribution of a Brownian motion that probably will result in a Log-normal distribution for prices.

The key step in most finance cases is generating increments of the driving Wiener Process (Brownian motion). A standard Brownian motion [Hull] with parameters (μ, σ) is a stochastic process $\{W_t : t \geq 0\}$ with following properties:

1. Its paths are continuous function of time t .
2. For $0 \leq t_1 < t_2 < \dots < t_m$, increments $W_{t_i} - W_{t_{i-1}}$, $i = \overline{1, m}$, a mutually independent.
3. $W_{t+s} - W_t \sim N(\mu, \sigma^2 s)$.
4. $W_0 = 0$.

In this case we can say that $W_t \sim B(\mu t, \sigma^2 t)$.

Moreover, the vector $(W_{t_0}, \dots, W_{t_m})$ is multivariate normal with mean μ and covariance matrix determined by $Cov[W_{t_i}, W_{t_j}] = \sigma^2 \cdot \min(t_i, t_j)$, $i, j = \overline{1, m}$.

μ is called the drift, and σ is called the volatility. When $\mu = 0, \sigma = 1$, we have a standard Brownian motion (SBM).

3.1 Generation BM paths using Forward Increments

3.1.1 Generation Paths for general stochastic process

Consider an underlying process that described by the stochastic differential equation (SDE)

$$(2) \quad dS = a(S; t)dt + b(S; t)dW$$

where $S(t)$ is the asset price and dW is the increment of a standard Brownian motion, with zero drift and unit volatility. In general, the drift coefficient a and the volatility b can depend on time t and on the value of the state variable S . The continuous time stochastic process S is called an Ito process; its mean and variance are:

$$E(dS) = a(S; t)dt, \quad \text{Var}(dS) = b^2(S; t)dt.$$

These process gives continuous solution trajectories (with probability one) and over time intervals short enough that a and b are nearly constant, the distribution of price changes is Normal with $\mu = 0, \sigma = 1$. The solution to the SDE is a random variable: at each time t , the solution value $S(t)$ has a specific probability distribution, and there is a certain covariance structure between the values at different times. We suppose that we are given a non-random starting value S_0 at time $t_0 = 0$. Our goal is to provide a rule for computing m random variables S_1, \dots, S_m , corresponding to a given list of n different times t_1, \dots, t_m , so that the random variables $\{S_j\}, j = \overline{1, m}$, have the same distribution structure as the samples $\{S(t_j)\}, j = \overline{1, m}$, of the true solution.

Given independent standard normal $\{Z_j\}, j = \overline{0, m}$, we can generate Brownian process by setting

$$(3) \quad W_{t_0} = \sqrt{t_0} \cdot Z_0, \quad W_{t_j} = W_{t_{j-1}} + \sqrt{t_j - t_{j-1}} \cdot Z_j, \quad j = \overline{1, m}.$$

Alternative way to generate $\{W_{t_j}\}, j = \overline{0, m}$, is to compute the Cholesky factorization of the covariance matrix and using it to multiply the column vector (Z_0, \dots, Z_m) .

We can construct an approximation to $S(t)$ at t_1, \dots, t_m as

$$(4) \quad S_{j+1} = S_j + a(S_j, t_j)(t_{j+1} - t_j) + b(S_j, t_j)\sqrt{t_j - t_{j-1}} \cdot Z_j, \quad j = \overline{0, m-1}.$$

According to Monte Carlo algorithm, we need to generate many paths to achieve a given accuracy. So, in practice, formula (4) corresponds to generating sample paths $S_j^i, i = \overline{1, N}$, where N is number of samples.

$$(5) \quad S_{j+1}^i = S_j^i + a(S_j^i, t_j)(t_{j+1} - t_j) + b(S_j^i, t_j)\sqrt{t_j - t_{j-1}} \cdot Z_j^i, \quad j = \overline{0, m-1}, \quad i = \overline{1, N}.$$

Once we have this collection of sample paths, we can estimate whatever summary statistics we want.

3.1.2 Generation Paths for Geometric Brownian motion (lognormal random walk)

Geometric Brownian motion (GBM) with parameters (μ, σ) is a stochastic process

$\{X_t : t \geq 0\}$ if $\ln(X_t) \sim B(\mu - \sigma^2/2) \cdot t, \sigma^2 t$, or write

$$X_t \sim LN((\mu - \sigma^2/2) \cdot t, \sigma^2 t).$$

For this process the SDE (1) takes $a(S, t) = \mu \cdot S$, $b(S, t) = \sigma \cdot S$. Therefore GBM is described by the SDE

$$(6) \quad \frac{dS}{S} = \mu \cdot dt + \sigma \cdot dW,$$

where the drift μ and volatility σ are constant parameters (μ is the (constant) instantaneous expected return on the stock, σ is the (constant) instantaneous standard deviation of stock returns.

The equation (6) is the most widely used model of stock price behavior.

This SDE has the explicit solution

$$(7) \quad S_t = S_0 \cdot \exp\left[\left(\mu - \frac{\sigma^2}{2}\right) \cdot t + \sigma \cdot W_t\right].$$

In this case, simulating values of S_t reduces to simulating values of W_t by formula (3). Generated values at a fixed time t of the underlying process is

$$(8) \quad S_t = S_0 \cdot \exp\left[\left(\mu - \frac{\sigma^2}{2}\right) \cdot t + \sigma \cdot Z\right], \quad Z \sim N(0,1).$$

Financial models that depend only on the underlying price at the time of maturity T (i.e. European options) will follow the above equation with $t = T$.

In case if a financial model depends on the price of some asset at certain specific times t_1, \dots, t_m approximations of the model in these intermediate times might be found.

Since the coefficients of (4) are constant then the exact solution of this across discrete times is

$$(9) \quad X_{j+1} = X_j + \left(\mu - \frac{\sigma^2}{2}\right) \cdot (t_{j+1} - t_j) + \sigma \cdot \sqrt{t_{j+1} - t_j} \cdot Z_j, \quad j = \overline{0, m-1}.$$

We can find S_j by setting $S_j = \exp(X_j)$, $X_0 = \ln(S_0)$.

As an example, we might want to hedge the position every day during 6 months depending on what the prices do that day; in that case you could set $m = 126$ (number of trading days in 6-month period).

3.1.3 Generation Paths for Geometric Brownian motion of multiple correlated assets

Now we generate values of multiple correlated assets, $\{S_t^{(i)}\}$, $i = \overline{1, k}$, where each $S_t^{(i)}$ is described by (4) with asset-specific parameters μ_i and σ_i , and driving Wiener process $W_t^{(i)}$. If $E[W_t^{(i)}, W_t^{(j)}] = \rho_{ij} \cdot t$, where ρ_{ij} is correlation between the Brownian Motions driving the assets $S_t^{(i)}$ and $S_t^{(j)}$, then

$$(10) \quad S_t^{(i)} = S_0^{(i)} \cdot \exp\left[\left(\mu_i - \frac{\sigma_i^2}{2}\right) \cdot t + \sigma_i \cdot \mathbf{Z}_i\right],$$

where $\mathbf{Z} = (Z_1, \dots, Z_k)$ is multivariate normal with mean zero and covariance matrix

$\Sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$. \mathbf{Z} can be generated, for example, through Cholesky factorization of Σ [Broadie at al.]. To obtain a discrete path of asset prices the above process can be repeated: the price of some asset at certain specific times t_1, \dots, t_m for each asset is given by

(11)

$$X_{j+1}^{(i)} = X_j^{(i)} + \left(\mu_i - \sigma_i^2 / 2 \right) \cdot (t_{j+1} - t_j) + \sigma_i \cdot \sqrt{t_{j+1} - t_j} \cdot Z_j^{(i)}, \quad j = \overline{0, m-1}, i = \overline{1, k}$$
 We can find $S_j^{(i)}$ by setting $S_j^{(i)} = \exp(X_j^{(i)})$.

3.1.4 Generation Paths of Term Structure models

Short rates are the short term interest rates that might happen in the future (i.e. random variables). Short rate process (a random process) can be described by the following SDE (Vasicek model specifies Ornstein-Uhlenbeck process) [[Broadie et al.](#)]

$$(12) \quad dr_t = \alpha(b - r_t) \cdot dt + \sigma \cdot dW_t$$

This simple model specifies a diffusion process for (instantaneous, continuously-compounded) short rate r_t (a short rate at time t). In the model (9) parameters α (the reversion speed), b (the reversion level), and σ (the volatility) are all constant. Term $\alpha(b - r_t) \cdot dt$ is a deterministic term (mean) with the drift $\alpha(b - r_t)$ and term $\sigma \cdot dW_t$ is stochastic term with the volatility σ . The mean-reverting form of the drift is an attractive feature in modeling interest rates. The equation (12) remain solvable if b varies with time as it is usually necessary to match an initial term structure. In this case, the solution is

$$(13) \quad r_t = e^{-\alpha t} \cdot r_0 + \alpha \int_0^t e^{-\alpha(t-s)} \cdot b(s) \cdot ds + \sigma \int_0^t e^{-\alpha(t-s)} \cdot dW_s.$$

Let denote $X(t) = \int_0^t e^{-\alpha(t-s)} \cdot dW_s$. The integrand $(e^{-\alpha(t-s)} \cdot dW_s)$ of this integral (which is stochastic) on the right is normally distributed with mean zero and variance

$$(14) \quad \int_0^t e^{-2\alpha(t-s)} \cdot ds = \frac{1}{2\alpha} (1 - e^{-2\alpha t}).$$

Therefore, simulating values of r_t reduces to simulating values from a normal distribution. The right mean and variance can be defined as

$$(15) \quad \mu(s, t) = \alpha \int_s^t e^{-\alpha(t-u)} b(u) \cdot du; \quad \sigma^2(s, t) = \frac{\sigma^2}{2\alpha} (1 - e^{-\alpha(t-s)}).$$

Given r_s , $s < t$, the interest rate at time t is normally distributed with mean $e^{-\alpha(t-s)} \cdot r_0 + \mu(t, s)$ and variance $\sigma^2(s, t)$.

The path is generated recursively by setting

$$(16) \quad r_{t_i} = e^{-\alpha(t_i - t_{i-1})} \cdot r_{t_{i-1}} + \mu(t_{i-1}, t_i) + \sigma(t_{i-1}, t_i) Z_i, \quad i = \overline{1, m},$$

where Z_1, \dots, Z_m are independent standard normal. The function b should be chosen to reproduce the initial term structure and $\mu(s, t)$ should be evaluated numerically.

To prevent this process from becoming negative (if $r_0 \geq 0$) and 0 if $(2\alpha b > \sigma^2)$ [Cox, Ingersoll, and Ross](#) (CIR) proposed add factor $\sqrt{r_t}$ in the diffusion coefficient

$$(17) \quad dr_t = \alpha(b - r_t) \cdot dt + \sigma \cdot \sqrt{r_t} \cdot dW_t.$$

Hull and White [[Hull and White](#)] (HW) model has a great level of analytical tractability and is currently popular. This model implies normally distributed interest rates and lognormal prices of the underlying process, and can be viewed as “Vasicek fitted to term structure” with the parameter α representing the speed of mean reversion.

The defining stochastic differential equation for this model is given by

$$(18) \quad dr_t = (\theta_t - \alpha \cdot r_t) \cdot dt + \sigma \cdot dW_t$$

The single time-dependent parameter in the drift that allows the model to fit the observed term structure is given by

$$(19) \quad \theta_t = \frac{\partial f(0, t)}{\partial t} + \alpha \cdot f(0, t) + \frac{\sigma^2}{2\alpha} (1 - e^{-2\alpha t}),$$

where $f(0; t)$ is the instantaneous forward rate.

The volatility structure of spot rates is therefore determined by both the volatility and rate of mean reversion of the reversion of the short rate:

$$(20) \quad \sigma_R(t, s) = \frac{\sigma}{\alpha(s-t)} \cdot (1 - e^{-\alpha(s-t)}).$$

To apply Monte Carlo simulation, let $B(t; T)$ = value of zero-coupon bond at time t , maturity T ,

$f(t, T) = -\frac{\partial}{\partial T}(\ln B(t, T))$ is the instantaneous forward rate at time t . Hull-White model is a

special case of one-factor HJM [[Heath, Jarrow, and Morton](#)] model with $\sigma(t, T) = \sigma \cdot e^{-\alpha(T-t)}$.

$$(21) \quad f(t, T) = f(0, T) + \int_0^T \mu(u, T) du + \int_0^T \sigma(u, T) dW(u),$$

where $\mu(t, T) = \sigma(t, T) \cdot \int_t^T \sigma(u, T) du$.

Therefore,

$$(22) \quad \int_0^T \mu(u, T) du = \sigma^2 \left(\frac{e^{\alpha(T-t)} - e^{-\alpha T}}{\alpha^2} - \frac{e^{2\alpha(T-t)} - e^{-2\alpha T}}{2\alpha^2} \right),$$

so $f(0, T) + \int_0^T \mu(u, T) du$ is a deterministic function that is the same for all simulated paths. So,

we actually need to simulate only the last term of the right side (21):

$$\int_0^T \sigma(u, T) dW(u) = \sigma \cdot e^{\alpha(T-t)} \cdot X(t), \quad X(t) = \int_0^T e^{-\alpha(t-u)} dW(u) \quad \text{is Markov, and}$$

therefore $X(t_i)$ $i = \overline{1, m}$, can be generated as

$$(23) \quad X(t_{i+1}) = e^{-\alpha(t_{i+1}-t_i)} \cdot X(t_i) + \frac{1 - e^{-2\alpha(t_{i+1}-t_i)}}{2\alpha} \cdot Z_i$$

Thus, using equations (21-23) the value of $f(t_k, T)$ in a particular path is computed as

(24)

$$f(t_k, T) = f(0, T) + \sigma^2 \left(\frac{e^{\alpha(T-t_k)} - e^{-\alpha T}}{\alpha^2} - \frac{e^{2\alpha(T-t_k)} - e^{-2\alpha T}}{2\alpha^2} \right) + \sigma \cdot e^{\alpha(T-t_k)} \cdot X(t_k)$$

In case of more complex models (i.e. volatility depends on time) the exact distribution of the underlying process is generally either unknown or too complicated to work with directly. If the randomness in the drift and volatility is limited to dependence on the current state of the process, making the process Markovian, the model can be described by vector SDEs of the form

$$(25) \quad dX_t = \mu(X_t, t) \cdot dt + \sum_{i=1}^m \sigma_i(X_t, t) \cdot dW_t^{(i)}.$$

If a process cannot be described by (24), its dynamics must depend on past history in complicated way, so simulation is likely to be difficult, if not impossible [[Broadie at al.](#)].

3.2 Generation BM paths using some variance-reduction techniques

Variance-reduction techniques serve to improve the statistical properties of a collection of simulated paths (Chapter 3.1) but not to improve the way any one path is generated. As we mention before, the accuracy of this simulation depends on the number of simulation paths, N . A large number of simulation runs is usually needed for reasonable accuracy. However, variance reduction techniques can be used to modify the original problem to reduce s and achieve reasonable accuracy with a smaller number of runs N .

Variance reduction methods work on exactly the same principle as that of hedging an option position that is the pay-off of a hedged portfolio will have a much smaller variability than an unhedged pay-off. This corresponds to the variance (a standard error) of a simulated hedge portfolio being much smaller than of the unhedged pay-off.

The most used in finance [[Broadie at al.](#)] is following variance-reduction techniques:

- Antithetic variates
- Control variates
- Importance Sampling
- Stratified sampling.
- Brownian Bridge
- Latin hypercube sampling.

3.2.1 Antithetic variates

Antithetic variates method is one of the most widely used because of its simplicity. Simulating with antithetic variates means that for each path $W_t = (W_{t_1}, \dots, W_{t_m})$ generated from a random sample (Z_1, \dots, Z_m) of independent standard normal numbers we generate a second path $\tilde{W}_t = (\tilde{W}_{t_1}, \dots, \tilde{W}_{t_m})$ from a random sample $(-Z_1, \dots, -Z_m)$, that is also from a standard normal distribution. Under this construction, \tilde{W} is just the mirror image of W . Based on N generated

sample paths, an unbiased estimator of the $\mu = E[W]$ is given by $\hat{\mu} = \frac{1}{N} \sum_{i=1}^N W_i$. Similarly,

each $\tilde{\mu} = \frac{1}{N} \sum_{i=1}^N \tilde{W}_i$ is also an unbiased estimator of μ . Therefore,

$\bar{\mu} = \frac{\hat{\mu} + \tilde{\mu}}{2}$ is also an unbiased estimator of μ .

$$\begin{aligned} \text{Var}\left[\frac{\hat{\mu} + \tilde{\mu}}{2}\right] &= \frac{1}{4} [\text{Var}(\hat{\mu}) + \text{Var}(\tilde{\mu}) + 2\text{Cov}(\hat{\mu}, \tilde{\mu})] = \\ &= \frac{1}{4m} [\text{Var}(W_t) + \text{Var}(\tilde{W}_t) + 2\text{Cov}(W_t, \tilde{W}_t)] = \text{Var}\left(\frac{\sum_{i=1}^{2m} W_{t_i}}{2m}\right) + \frac{\text{Cov}(W_t, \tilde{W}_t)}{2m} = \\ &= \frac{1}{2m} [\text{Var}(W_{2t}) + \text{Cov}(W_t, \tilde{W}_t)], \quad W_{2t} = (W_{t_1}, \dots, W_{t_{2m}}). \end{aligned}$$

The intuition behind the antithetic variates method is that the random inputs obtained from the collection of antithetic pairs $(Z_i, -Z_i)$ are more regularly distributed than a collection of $2N$ independent samples. Antithetic sampling makes the mean path always equals 0, whereas the mean over finitely many independent samples is almost surely different from 0.

A price of underlying asset can be obtained by averaging the values from $2m$ paths: $W_t^{(1)}, \dots, W_t^{(m)}$ and their mirror images $\tilde{W}_t^{(1)}, \dots, \tilde{W}_t^{(m)}$.

Antithetic sampling effective if

- $\text{Cov}(W_t, \tilde{W}_t) < 0$, this way the values computed from a path and its mirror image are negatively correlated.
- f is monotone in each of argument. Particular, if f is linear then the antithetic estimate of $E[f(Z_1, \dots, Z_m)]$ has zero error.
- f is symmetric. In this case the variance of an antithetic sample of size $2m$ is the same as an independent sample of size m .

3.2.2 Control variates

This method adjusts the estimate computed from a set of paths rather than the paths themselves. The adjustment is made based on analytically available information about the

underlying assets or about prices of simpler instruments. The basic idea of control variates is to replace the evaluation of an unknown expectation with the evaluation of the difference between the unknown quantity and another expectation whose value is known. The standard Monte-Carlo estimate of $\mu = E[W_t] = E[f(Z_t)]$ is $\frac{1}{N} \sum_{i=1}^N W_i$. If we know $\tilde{\mu} = E[\tilde{W}_t] = E[g(Z_t)]$

with estimate of $\tilde{\mu}$ is given by $\frac{1}{N} \sum_{i=1}^N \tilde{W}_i$, then the method of control variates uses the known error $\frac{1}{N} \sum_{i=1}^N \tilde{W}_i - \tilde{\mu}$ to reduce the unknown error $\frac{1}{N} \sum_{i=1}^N W_i - \mu$.

The controlled estimator has the form

$$(26) \quad \frac{1}{N} \sum_{i=1}^N W_i - \beta \left(\frac{1}{N} \sum_{i=1}^N \tilde{W}_i - \tilde{\mu} \right).$$

Since the term in parentheses has expectation zero, equation (26) provides an unbiased estimator of μ as long as β is independent. In practice, if the function $g(Z_t)$ provides a close approximation of $f(Z_t)$, we usually set $\beta = 1$ to simplify the calculation. The estimator with the smallest

variance is given by $\beta = \beta_* = \frac{\text{Cov}(W_t, \tilde{W}_t)}{\text{Var}(\tilde{W}_t)}$.

On practice, usually these variances and covariances are not known priori, but they can be estimated from the observed sample data as

$$(27) \quad \hat{\beta}_* = \frac{\sum_{j=1}^m (W_j - \bar{W})(\tilde{W}_j - \bar{\tilde{W}})}{\sum_{j=1}^m (\tilde{W}_j - \bar{\tilde{W}})^2},$$

where \bar{W} , $\bar{\tilde{W}}$ are the sample means of W_t , \tilde{W}_t , respectively. Variance of the process can be found as

$$(28) \quad \text{Var}[W_t(\beta_*)] = \text{Var}(W_t) - \frac{\text{Cov}^2(W_t, \tilde{W}_t)}{\text{Var}(\tilde{W}_t)}, \quad W_t(\beta_*) = W_t - \beta_*(\tilde{W}_t - E[\tilde{W}_t]).$$

In the interest rate setting it is natural to use bond prices as controls, subtracting the term

$$(29) \quad \beta \cdot \left(\frac{1}{N} \sum_{i=1}^N \exp \left(- \sum_{j=1}^m (t_j - t_{j-1}) \cdot r_{j-1}^{(i)} \right) - P_T \right)$$

from the payoff of an interest rate derivative and estimating the optimal β analogically (26).

If $t_j - t_{j-1} = \Delta t$, $\forall j = \overline{1, m}$, equation (29) will simplify to

$$(30) \quad \beta \cdot \left(\frac{1}{N} \sum_{i=1}^N \exp \left(- \Delta t \cdot \sum_{j=1}^m r_{j-1}^{(i)} \right) - P_T \right), \quad T = m \cdot \Delta t.$$

The control variates technique is efficient when there are some easily computed quantities that are highly correlated with the object of interest. This method is especially good for pricing Asian options depending on geometric averaging. In this case we can take the geometric average as a control variate to price Asian options which use arithmetic averaging. This technique is effective for it because analytical solutions for geometric case are often exact (depending on exactly how the averaging is defined). We will calculate a difference in value of the two options when they are evaluated along the same sample path. The control variates technique (see [Hull and White](#)) could also be successfully used when we deal with two similar options: a simpler option problem which has an analytic solution, and uses that solution to improve the accuracy of the more complex problem at hand (that has no analytic solution). Suppose we want to estimate numerically the value of option A that has no analytic solution (e.g., the American put option without dividends). We can identify a similar option B for which an analytic solution is available (e.g., a European put without dividends). We can run two simulations in parallel using the same Z and Δt to obtain estimates of the values F_A and F_B of options A and B , F_A^* and F_B^* , respectively. A more accurate estimate of the value of option A is then obtained as follows: $F_A = F_A^* - F_B^* + F_B$.

3.2.3 Importance sampling

The technique builds on the observation that an expectation under one probability measure can be expressed as an expectation under another through the use of a likelihood ratio. The intuition behind the method is to generate more samples from the region that is more important to the practical problem at hand.

Suppose we want to estimate $\theta = E_f [h(X)]$, where X has PMF (if X is a discrete random variable, PDF if X is continuous) $f(\cdot)$. Let $g(\cdot)$ be another PMF (or PDF) with the property that $g(x) \neq 0$ whenever $f(x) \neq 0$, and Y has PMF (PDF) $g(\cdot)$. Other words, g has the same support as f . Then

$$\theta = E_g \left[\frac{h(X)f(X)}{g(X)} \right].$$

If we set $h^*(X) := \frac{h(X)f(X)}{g(X)} \Rightarrow \theta = E_g [h^*(X)]$, where g is the importance

sampling PMF (PDF). An estimate of θ and variance of the estimate are given

$$(31) \quad \theta \approx \hat{\theta}_N = \frac{1}{N} \sum_{i=1}^N h^*(Y_i); \quad \text{Var}[\hat{\theta}_N] \approx \hat{S}_N = \frac{1}{N-1} \cdot \sum_{i=1}^N [h^*(Y_i) - \hat{\theta}_N]^2.$$

Variance reduction could be achieved by choosing h with the structure somewhat similar to $f \cdot g$. In general, h should be large where f and g are, so we can concentrate our sample points in locations with big contribution to the final result. From the other hand, one can be efficient using importance sampling to make rare events less rare. The example is a deep out-of-the-money option. Here we can use a probability distribution corresponding to an artificially large value of the drift parameter so as to generate an artificially high probability of expiring in the money ([Boyle](#), [Broadie et al.](#)). Importance sampling technique is also efficient with eliminating singularities in the objective function, or with reduction of a problem with infinite variance to one with finite one.

The most problematic cases for Monte Carlo based option pricing are options for which the probability of an occurrence of a strictly positive payoff is very small. Then we will get price and

variance estimates based on a few positive samples if we hit the payoff region or we get a zero payoff and variance if this improbable event does not occur. However in both cases we will get a very high relative error. More accurate results may be calculated by applying importance sampling to these options.

The most difficult aspect to importance sampling is in choosing a good sampling density, g . In general, one needs to be very careful for it is possible to choose g according to some good heuristic such as the maximum principle, but to then find that g results in a variance *increase*. In fact it is possible to choose a g that could result in an importance sampling estimator that has an infinite variance! This situation would typically occur when g puts too little weight relative to f on the tails of the distribution.

3.2.4 Stratification

Like many variance reduction techniques, stratified sampling seeks to make the inputs to simulation more regular than the random inputs. In stratified sampling, rather than drawing W_i randomly and independent from a given distribution, the method ensures that fixed fractions of the samples fall within specified ranges.

For example, we want to generate N m -dimensional random vectors from a given distribution $F(\cdot)$ for simulation input. The empirical distribution of an independent sample (W_1, \dots, W_N) will look only roughly like the true density. Stratified sampling can be used to ensure that there is the only one observation W_i^k lies between the $\frac{i-1}{N}$ and $\frac{i}{N}$ quantiles ($i = \overline{1, N}$) of the k -th marginal distribution for each of the m components. One way to implement that is to generate $N \cdot m$ independent uniform random numbers U_i^k on $[0,1]$, $k = \overline{1, m}$; $i = \overline{1, N}$ and set

$$W_i^k = F^{-1} \left[\frac{i + U_i^k - 1}{N} \right], \quad i = \overline{1, N}.$$

In order to achieve satisfactory sampling results, a good numerical procedure to calculate F^{-1} is needed.

An alternative is to apply the stratification only to the most important components (directions), usually associated to the eigenvalues of largest absolute value.

The obvious advantage of stratified sampling is that leads to a variance reduction. But in practice, stratified sampling method quickly become infeasible as the dimension k increases, unless N is made so small as to eliminate the effect on variance [Broadie at al.]. Generating a full stratified sample in this way requires generating N^k points that is not viable if the dimension large than 5 or 6. But using just the initial portion of the full set of N^k points would leave a large part of the unit hypercube completely unsampled, that give a poor results. Also to reach a good estimation, the number of sub-intervals on each dimension should be large enough (e.g. N greater than 30). But, on the other hand, if N is large however, and each simulation run is computationally expensive, then it may be the case that a lot of effort is expended in trying to estimate the optimal N_i 's.

3.2.5 Latin Hypercube Sampling (LHS)

Latin hypercube sampling [[Broadie at al.](#)] is an alternative to full stratification in high dimensional problems. It generates N of the N^k full-stratification bins in a particularly effective way. Those N points in dimension k are chosen so that all k marginal distributions are perfectly stratified.

The Latin Hypercube Sampling method was first introduced by [McKay at al.](#). To generate a Latin hypercube sample of size N in dimension k we can imply the algorithm:

1. Generate a $(N \times k)$ matrix of independent and uniform over $[0,1]$ numbers $U_j^i, j = \overline{1, k}, i = \overline{1, N}$.
2. Compute a vector $\pi_j, j = \overline{1, k}$, independent random permutations of $\{1, 2, \dots, N\}$ sampled uniformly from all $N!$ such permutations.
3. Calculate $V_j^i = \frac{\pi_j(i) - 1 + U_j^i}{N}, j = \overline{1, k}$.
4. The N points $V^i = (V_1^i, \dots, V_k^i), i = \overline{1, N}$, constitute a Latin hypercube sample of size N in dimension k .

Each of these vectors can be transformed into vectors of given distribution. In the Latin Hypercube Sampling method, the range of probable values for each component U_j^i is divided into N segments of equal probability. For example, for the case of dimension $k = 3$ and $N = 10$ segments, the parameter space is divided into $10 \times 10 \times 10 = 10^3$ cells. The next step is to choose 10 cells from the 10^3 cells. First, the uniform random numbers are generated to calculate the cell number. The cell number indicates the segment number the sample belongs to, with respect to each of the parameters. For example, a cell number (2, 8, 4) indicates that the sample lies in the segment 2 with respect to first parameter, segment 8 with respect to second parameter, and in segment 4 with respect to the third parameter. At each successive step, a random sample is generated, and is accepted only if it does not agree with any previous sample on any of the segment numbers.

As an advantage of LHS method is that it asymptotically eliminates the contribution to the variance due the additive part of the function being integrated. Disadvantage of this method is it does not provide a means of estimating a standard error to be used to form a confidence interval around a point estimate. Because the sample points are so highly correlated in hypercube sampling, proper estimation of error requires careful batching of the sample data. So, to obtain a good estimate of the error we should relax some of the variance reduction properties.

3.2.6 Generation BM paths using Brownian Bridge

Techniques for reducing the effective dimension (Brownian Bridge (BB) and the principal components (PC) can greatly improve the performance of QMC methods (see [Acworth], [Caflich], [Morokoff]). The idea of BB was first used in finance by Caflich.

The idea of BB construction lays in generation a discrete Wiener path $(W_{t_1}, \dots, W_{t_k})$ by sampling the terminal value first and then filling in the rest of the path recursively.

1. Generate $Z_i \sim N(0,1)$, $i = \overline{0, k-1}$, from uniform pseudo-random or quasi-random numbers (Chapter 2).
2. Set $W_{t_k} = \sqrt{t_k} \cdot Z_0$, $W_0 = 0$.
3. Calculate W_{t_i} , $i = \overline{0, k-2}$, recursively using the formula:

$$(32) \quad W_{t_{i+1}} = \left(\frac{t_k - t_{i+1}}{t_k - t_i} \right) \cdot W_{t_i} + \left(\frac{t_{i+1} - t_i}{t_k - t_i} \right) \cdot W_{t_k} + \sqrt{\frac{(t_{i+1} - t_i)(t_k - t_{i+1})}{(t_k - t_i)}} \cdot Z_{i+1}.$$

This construction follows from the Brownian Bridge properties:

- The distribution of $W_{t_{i+1}}$ conditional on W_{t_i} and W_{t_k} is normal with the mean that lies on the line segment connecting W_{t_i} and W_{t_k} .
- A standard deviation equal to the factor multiplying Z_{i+1} in (32).

The variance of the random part of the Brownian Bridge formula (32) is less than in Brownian motion formula (5) since $\sqrt{\frac{(t_{i+1} - t_i)(t_k - t_{i+1})}{(t_k - t_i)}} \leq \sqrt{(t_{i+1} - t_i)}$. Because much more of the

variance is contained in the first few steps of the Brownian bridge discretization due to the reduction in variance in the BB formula this results in reduction of effective dimension of the random walk simulation which increases the accuracy of Quasi-Monte Carlo methods.

Brownian Bridge discretization (BBD) is another method that is based on formula (32). The algorithm to determine the path $(W_{t_1}, \dots, W_{t_k})$ as follow (for convenience we assume that k is a power 2):

1. Generate $Z_i \sim N(0,1)$, $i = \overline{0, k}$, from uniform pseudo-random or quasi-random numbers (Chapter 2).
2. Denote $B(t_k) = W_{t_k} = \sqrt{t_k} \cdot Z_0$, $W_0 = 0$.
3. In formula (32) use Z_1 to generate the value at the midpoint $B(t_{[k/2]})$, Z_2 for $B(t_{[k/4]})$, Z_3 for $B(t_{[3k/4]})$, etc.

This can be done easily since for $u < v < w$, the distribution of $B(v)$ given $B(u)$ and $B(w)$ is normal with parameters depending only on u , v , w . The reason why this can be helpful for QMC is that by generating the Brownian motion path in this way, more importance is given to the first few uniform numbers, and thus the effective dimension of a function depending on $B(t_1), \dots, B(t_k)$ should be decreased by doing that.

Also we can decompose the variance-covariance matrix of the prices to be simulated using principal components, and then generate the prices using this decomposition [[Acworth](#)]. An advantage of this approach over BB is that it can be used to generate prices of correlated assets whereas BB can only be applied for generating prices coming from a single path. However, PC requires more computation time than BB for the generation of the prices. But using techniques from [[Akesson](#)] PC methods could be speed up.

But the Brownian Bridge is not a panacea. It has several disadvantages. It can demand additional computation complexity/time (reordering the path) for path-dependent problems. [Papageorgiou](#) also shows that the Brownian Bridge does not offer a consistent advantage in Quasi-Monte Carlo Integration for lognormally distributed asset prices. Covariance matrix decomposition can be interpreted as a change to the integrand or to the sample points. Such a change may yield a harder problem relative to a fixed set of sample points.

3.3 Demonstration of BM paths for different variance-reduction techniques

The graphs below are used for evaluation of European Call Option with different variance reduction techniques with $N=50$ (number of paths) and $s=12$ (Number of dimension -time-intervals). Faure sequences with base 13 were used.

Fig. 16 Without any variance-reduction techniques

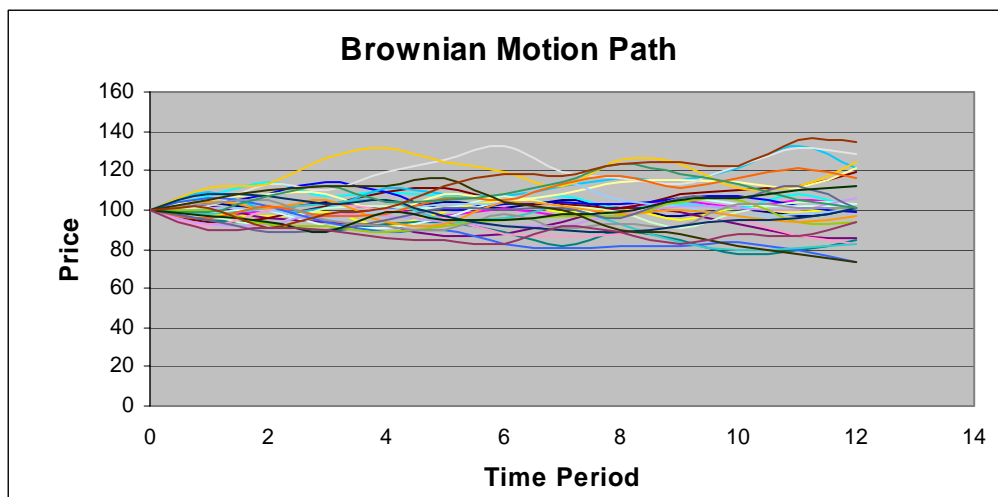


Fig. 17 With Brownian Bridge

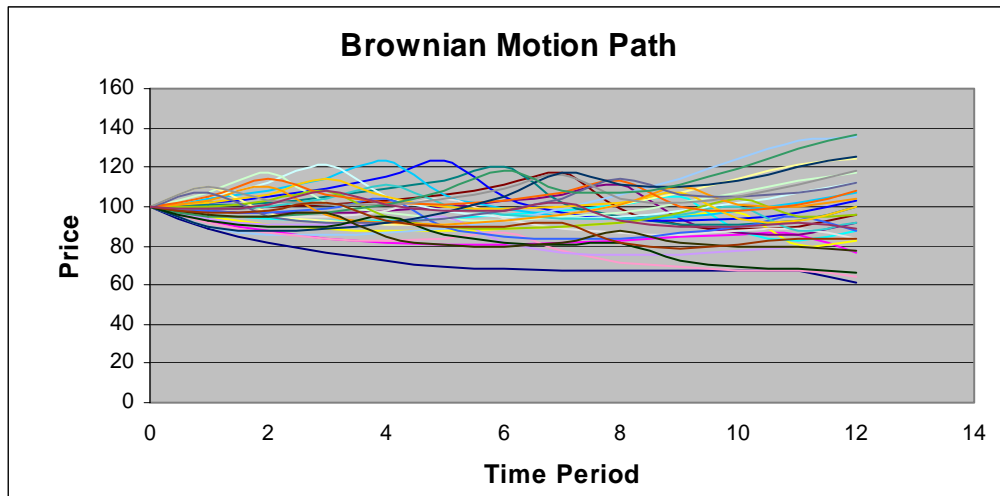


Fig. 18 With Brownian Bridge Discretization

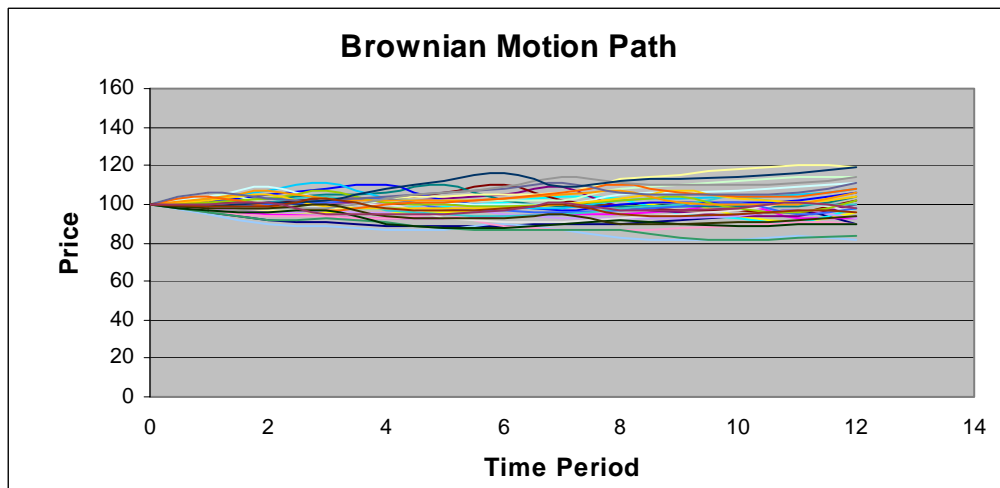
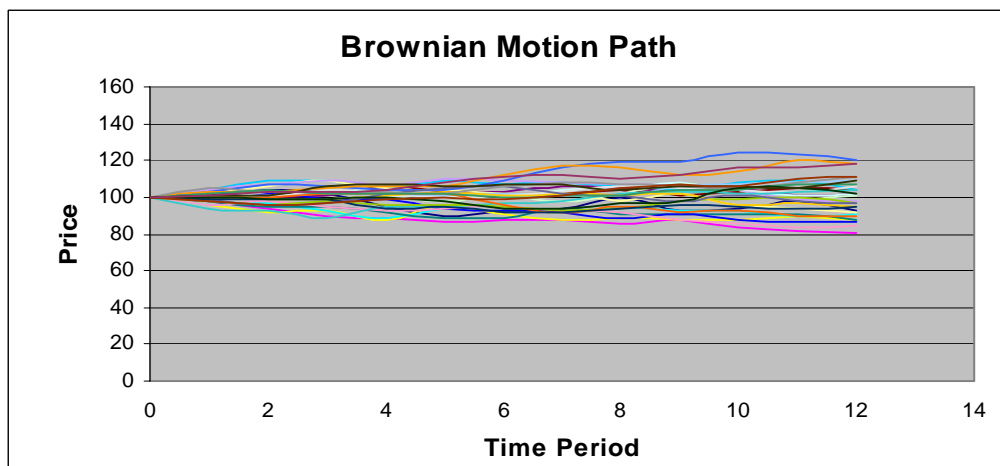


Fig. 19 With Latin Hypercube technique



We can observe from the graphs above that Brownian Bridge technique allow us to reach smaller variance between the paths. Influence of other variance reduction techniques as well as above techniques will be demonstrated more while evaluating path-dependent financial securities in next chapter.

CHAPTER 4

APPLICATION TO FINANCE PROBLEMS

Simulation is a powerful tool for generating the relevant risk-neutral probability distributions that allow discounting of expected terminal option values at the risk-free rate. Simulation is rather simple and flexible in that it can handle complex derivative structures, can be easily modified to accommodate different (even complex) underlying stochastic processes (e.g., mean reversion, jump, or mixed jump-diffusion processes), or can be used when the sample security prices are drawn from an empirical distribution. MC (especially QMC) simulation is especially effective for valuing (European) path-dependent options where the option payoff depends on the history of the underlying state variable and not just its terminal value (e.g., lookback options that depend on the maximum or minimum price reached during the option's life, or Asian options that depend on the average price achieved). The procedure can also be successfully extended to cases where the payoff from the derivative depends on several underlying market variables. It also can handle complex payoffs and complex stochastic processes. MC and QMC simulation tends to be even more numerically efficient than other procedures (as trees and finite difference methods etc.) when there are three or more stochastic variables. This is because the running time, even for MC simulation, increases approximately linearly [Hull] with the number of variables, whereas in most other procedures it increases exponentially. It also has the advantage that it can provide confidence limits via the standard error of the estimated option value. One of disadvantages of MC (QMC) approach is it is not naturally suited to valuing American-style options. There are two ways to adapt this method to handle the options [Hull]. One of them uses a least-squares analysis to relate the value of continuing (i.e. not exercising) to the values of relevant variables. Another one is parameterization the early exercise boundary and determination it iteratively by working back from the end of the life of the option to the beginning.

For example, in risk-neutral world, it takes several steps to evaluate the price S of the security:

- Generate a matrix of random paths for S from a risk-neutral world. It can be done with techniques from Chapter 3.
- Calculate the payoff for each path from the security (here we should use appropriate payoff formulas for the security).
- Calculate the mean of the sample payoffs to obtain an estimate of the expected payoff in a risk-neutral world.
- Discount the expected payoff at the risk-free rate to get an estimate of the value of the security.

4.1 Evaluating European options

A *European call (put) option* gives the holder the right to buy (sell) the underlying asset with current price S_0 by a certain date (T is a maturity or expiration date) for a certain price (K is the strike price). Suppose S_T is the final price of the asset at maturity. The payoff from a long position in a European call option is $\max(S_T - K, 0)$. The payoff to the holder of a short position in the European call option is $\min(K - S_T, 0)$. The payoff from a long position in a European put option is $\max(K - S_T, 0)$. The payoff to the holder of a short position in the European put option is $\min(S_T - K, 0)$.

The European call option solved by Monte Carlo simulation is given by the integral below:

$$C(S_t, t) = e^{-r(T-t)} E_t[\text{payoff}(S_T)] \Rightarrow$$

$$C(S_t, t) = e^{-r(T-t)} \int_0^{\infty} \max(S_T - K, 0) * f(S_T, T / S_T, t) dS_T$$

$$S_T = S_0 \cdot \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma \cdot Z \sqrt{T}\right], \quad Z \sim N(0,1).$$

$$\Rightarrow C(S_t, t) = e^{-r(T-t)} \int_0^{\infty} f(S_T) dS_T.$$

Where $E_t[.]$ is the payoff expectation under risk-neutral (or martingale) measure conditional to the information at the (current) time t ; $f(S_T, T / S_T, t)$ is the *risk-neutral* transition density function of S , which is lognormal for the geometric Brownian motion (GBM); S_t is the value of the underlying asset at the (current) instant t , and $\mu = r$ for a non-dividend paying stock, r is the risk-free interest rate, σ is a volatility. We can easily apply Monte Carlo (QMC) technique to estimate a price of the European option under Geometric Brownian motion as follows

1. Calculate payoff for each path ($i = \overline{1, N}$), where N is number of simulations. For example, for a European call the payoff along the i -th path is:

$$C_T^{(i)} = \max[S_T^{(i)} - K, 0],$$

where

$$S_T^{(i)} = S_0 \cdot \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma \cdot Z_i \sqrt{T}\right],$$

$\mu = r$ - for a non-dividend paying stock.

2. Take an average of those payoffs: $\hat{C}_T = \frac{1}{N} \cdot \sum_{i=1}^N C_T^{(i)}$.

3. Discount this value by the risk-free rate to get the price of the option: $C = e^{-rT} \cdot \hat{C}_T$.

4. Compute the standard error: $SE(\hat{C}) = \frac{\sqrt{\frac{\exp(-2rT)}{N-1} \cdot \sum_{i=1}^N (C_T^{(i)} - \hat{C}_T)^2}}{\sqrt{N}}$,

$$\left(\text{Var}(C_T) \approx \text{Var}(\hat{C}) = \frac{\exp(-2rT)}{N-1} \cdot \sum_{i=1}^N [C_T^{(i)} - \hat{C}_T]^2 \right).$$

The exact solution for the prices at time zero of a European call option on a dividend-paying stock is given the Black-Scholes pricing formula [Hull]:

$$C = S_0 e^{-\delta T} \cdot N(d_1) - K \cdot e^{-rT} N(d_2), P = K \cdot e^{-rT} N(-d_2) - S_0 e^{-\delta T} \cdot N(-d_1);$$

$$d_1 = \frac{\ln(S_0 / K) + (\mu + \sigma^2 / 2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}, \quad \mu = r - \delta.$$

δ is a dividend yield.

The Table below shows the number simulations needed for different methods to achieve the price with error $\varepsilon = 0.02$. For Monte Carlo method this value can be found as $N > \left(\frac{2\text{Var}(\hat{C})}{\varepsilon} \right)^2$.

Note that to compute European option under distribution other than GBM, we should divide the time period $(0, T)$ into M equal intervals such that $\Delta t = T / M$ and then generate values of S_t at the end of these intervals as follows

$$S_t^{(i)} = e^{-x_t(i)}, \quad x_t(i) = x_t(i-1) + (\mu - \sigma^2 / 2) \cdot \Delta t + \sigma\sqrt{\Delta t} \cdot Z_i,$$

$$t_i = i \cdot \Delta t, i = \overline{1, M}.$$

To reduce a variance we can implement *antithetic variates* technique (see Chapter 3.2.1). For example, for a European call the implementation of this technique as follow:

1. Calculate two payoffs for each path ($i = \overline{1, N}$), where N is number of simulations:

$$C_T^{(i)} = \max[S_T^{(i)} - K, 0] = \max\left(S_0 \cdot \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma \cdot Z_i \sqrt{T}\right] - K, 0 \right)$$

and the payoff to the option on the perfectly negatively correlated asset as

$$\bar{C}_T^{(i)} = \max\left(S_0 \cdot \exp\left[\left(\mu - \frac{\sigma^2}{2}\right)T + \sigma \cdot (-Z_i) \cdot \sqrt{T}\right] - K, 0 \right).$$

$\mu = r$ - for a non-dividend paying stock.

2. Take an average of those payoffs: $\hat{C}_T = \frac{1}{N} \cdot \sum_{i=1}^N \frac{C_T^{(i)} + \bar{C}_T^{(i)}}{2}$.
3. Discount this value by the risk-free rate to get the price of the option: $C = e^{-rT} \cdot \hat{C}_T$.
4. Compute the standard error: $SE(\hat{C}) = \frac{\sqrt{\frac{\exp(-2rT)}{N-1} \cdot \sum_{i=1}^N (C_T^{(i)} - \hat{C}_T)^2}}{\sqrt{N}}$.

This way we obtain a much more accurate estimate from N pairs of $(C_T^{(i)}, \bar{C}_T^{(i)})$ than from $2N$ of $(C_T^{(i)})$, and is also computationally cheaper (we need less computational operations). This method also ensures that the mean of the normally distributed samples Z is exactly zero which also helps improve simulation.

As an practical implementation, suppose $S_0 = 100$, $K = 100$, $T = 1$, $r = 5\%$, $\delta = 2\%$, $\sigma = 0.10$.

Table below represent the results of evaluating the European call option by different techniques and compare them with exact solution. Sobol sequences were used to generate uniform samples in QMC method.

Table 11. Comparison of different variance reduction techniques for European Call ($N=1000$, $s=12$)

	Price	Standard Error	Computation time (sec)
<i>Black-Scholes solution</i>	5.4713	-	-
<i>Simple MC</i>	5.6539	0.0334	6
<i>Simple QMC (none of VR techniques used)</i>	5.3367	0.0246	8
<i>Antithetic Variates</i>	5.3531	0.0216	12
<i>Control Variates *</i>	5.4307	0.0074	8
<i>Control Variate* with Antithetic Variate</i>	5.4471	0.0044	12
<i>Latin Hypercube with BBD</i>	5.4528	0.0034	8
<i>Brownian Bridge</i>	5.4485	0.0042	5
<i>Brownian Bridge Discretization</i>	5.4485	0.0042	8

*For this case Delta is used as Control variate

Using QMC method we could decrease the standard error comparing to simple Monte Carlo, and this difference rises with increasing number of paths. For example, in case $N=10,000$ and $s=1$ (that is enough to evaluate European option) the standard error given by QMC method is

SE=0.0054 whereas this error is more than twice as higher with standard Monte Carlo: SE=0.0132. Also the standard error can be reduced by using some variance-reduction techniques. Thus, the implementation of Brownian Bridge techniques allows us to significantly decrease the standard error and computation time (approximately by 8 times comparing to simple Monte Carlo and by 5.9 times comparing to QMC method). Also, the variance could be reduced by using the addition of Latin hypercube and Brownian Bridge Discretization techniques.

Notice: Computational time in Table 11-Table 14 is relative time that includes time spent on transfer simulated stock prices to Excel spreadsheet and using them to evaluate the derivative. In case if the derivative is evaluated using a matrix of simulated prices in Visual Basic without transferring it to Excel, the computational time would be significantly less.

4.2 Evaluating Barrier Options

Barrier option is an option whose payoff depends on whether the path of the underlying asset has reached a barrier (i.e. a certain predetermined level). Barrier options can be classified as either knock-out options or knock-in options [Hull]. A knock-out (down-and-out and up-and-out) option ceases to exist when the underlying asset price reaches a certain barrier; a knock-in option comes into existence only when the underlying asset price reaches a barrier (down-and-in and up-and-in). For example, a *down-and-out call* is a type of knock-out option that ceases to exist if the asset price falls below a certain barrier level, H . The barrier level is below the initial asset price. A *down-and-in call* is a type of knock-in option that comes into existence only if the asset price falls below the barrier level, H . The sum of the prices of a down-and-out call and down-and-in call gives us the value of a regular European call: $C_T = C_{do,T} + C_{di,T}$. A down-and-out is a put option that ceases to exist when a barrier less than the current asset price is reached. A down-and-in put is a put option that comes into existence only when the barrier is reached.

Barrier option is an example of path-dependent derivatives [Clewlow at al]. If we assume that the underlying asset price is checked continuously for the crossing of the barrier, then we can apply analytical formulas to price it [Hull, p.439]. But if the underlying asset prices are checked (fixed) at most once a day or even less frequently then the price of the option will significantly be affected because the price is much less likely to be observed crossing the barrier in this case. So, implementing of analytical formulas becomes problematical, due to increasing complexity of the formulas. This option can be easily priced by Monte Carlo (or QMC) simulation.

Let assume that the asset price follows GBM then the simulation of the asset is given by

$$S_{j+1} = S_j \cdot \exp(\mu \cdot \Delta t + \sigma \cdot \sqrt{\Delta t} \cdot Z_j), \quad j = \overline{0, m},$$

where Δt is one day (for example) and $Z \sim N(0,1)$. The Monte Carlo simulation proceeds in exactly the same way as for standard option, except that at time step we check whether the asset price has crossed the barrier level H . If so then for down-and-out call option, for example, we terminate the simulation of that path and the pay-off for that path is zero. The value of the option for each path is given by

$$(33) \quad C^{(i)} = \begin{cases} 0, & \text{if } \exists j: S_j^{(i)} \leq H \quad (j = \overline{1, m}), \\ e^{-rT} \cdot \max(S_T - K, 0), & \text{if } S_j^{(i)} > H, \forall j = \overline{1, m}; \quad i = \overline{1, N}. \end{cases}$$

The “true” value of the option over N simulated paths is

$$(34) \quad C_{do,T} = C_T = \sum_{i=1}^N C^{(i)} \cdot \mathbf{I}_{\{S^{(i)} > H\}}, \mathbf{I}_{\{S^{(i)} > H\}} = \begin{cases} 1, S^{(i)} > H \\ 0, S^{(i)} \leq H \end{cases}, i = \overline{1, N}.$$

The price for an *up-and-out call* option is similar to the price of down-and-out option with difference that we terminate the simulation of a path when the price on a step become equal or greater the barrier level and the pay-off for that path is zero.

An important issue for barrier options is the frequency that the asset price is observed that used to investigate whether the barrier has been reached. As it was mentioned before, the analytic formula (Hull) assume that the price is observed continuously and sometimes this is in case, but usually the price is observed daily. To adjust the analytical formulas to price a barrier option where the price of underlying asset observed discretely, [Broadie, Glasserman, and Kou](#) proposed to replace the barrier level H by $H \cdot \exp(0.582 \cdot \sigma \cdot \sqrt{T/m})$ for an up-and-out (in) options and by $H \cdot \exp(-0.582 \cdot \sigma \cdot \sqrt{T/m})$ for down-and-out (in) options.

Suppose we want to evaluate a Down-and-Out Barrier option with following parameters: $S_0 = 100, K = 100, T = 1, r = 5\%, \delta = 2\%, \sigma = 0.10, H = \99 .

Table below represent the results of evaluating the Barrier call option by different techniques and compare them with exact solution. Sobol sequences were used to generate uniform samples in QMC method.

Table 12. Comparison of different variance reduction techniques for Down-and-Out Call ($N=1000, s=12$)

	Price	Standard Error	Computation time (sec)
<i>Black-Scholes solution</i>	3.0774	-	-
<i>Simple MC</i>	3.3869	0.1006	8
<i>Simple QMC (none of VR techniques used)</i>	3.2766	0.0647	5
<i>Antithetic variate</i>	3.2675	0.0618	11
<i>Brownian Bridge</i>	3.1648	0.0284	5
<i>Latin Hypercube with BB</i>	3.1325	0.0179	7

Using QMC method we could decrease the standard error comparing to simple Monte Carlo (approximately by 1.55 times), and this difference rises with increasing number of paths and implementing of some variance-reduction techniques. For example, the implementation of Brownian Bridge techniques allows us to significantly decrease the standard error (approximately by 3.54 times) and computation time. To achieve this reduction with the simple Monte Carlo would require increasing the number of simulations by $3.54 \times 3.54 = 12.53$ times that would result in significant increase in the computation time. The addition of the Latin hypercube and Brownian Bridge techniques result s in even more variance-reduction: the standard error decreases approximately by 5.62 times comparing to simple Monte Carlo method and by 1.59 times comparing to Brownian Bridge techniques.

4.3 Evaluating Asian Options

Asian option [Hull] is an option with a payoff dependent on the average price of the underlying asset during a specified period.

The payoff from an *average price call* is $C_{ave} = \max(0, S_{ave} - K)$ and the payoff from an *average price put* is $P_{ave} = \max(0, K - S_{ave})$, where S_{ave} is the average value of the underlying asset calculated over life of option. Asian options are less expensive than regular options and are arguably more appropriate than regular option for meeting some of the needs of risk management. The payoff from an *average strike call* is $C_{ave}^{(Strike)} = \max(0, S_T - S_{ave})$ and from an *average strike put* is $P_{ave}^{(Strike)} = \max(0, S_{ave} - S_T)$. Average strike options can guarantee that the average price paid (received) for a asset in frequent trading over a period of time is not greater (not less) than the final price. If Asian options are defined in term arithmetic averages $S_{ave} = A_T = \frac{1}{m} \sum_{j=1}^m S_j$, exact analytic pricing formulas are not available. We can easily apply

Monte Carlo (QMC) methods to price this derivative:

1. For each path we find $S_{ave}^{(i)} = A_T = \frac{1}{m} \sum_{j=1}^m S_j^{(i)}$;
2. Calculate payoff of the i -th path. For example, for an average price call the payoff is $C_{ave}^{(i)} = \max(0, S_{ave}^{(i)} - K)$, $i = \overline{1, N}$.
3. Discount this value by the risk-free rate to get the price of the option: $\hat{C}_{ave}^{(i)} = e^{-rT} \cdot C_{ave}^{(i)}$.
4. Take an average of those payoffs: $\hat{C}_{ave} = \frac{1}{N} \cdot \sum_{i=1}^N \hat{C}_{ave}^{(i)}$.
5. Compute the standard error: $SE(\hat{C}_{ave}) = \frac{\sqrt{\frac{1}{N-1} \sum_{i=1}^N (\hat{C}_{ave}^{(i)} - \hat{C}_{ave})^2}}{\sqrt{N}}$.

If the average price found as geometrical average then there are analytical formulas for valuing European average price option (since the price of the asset is assumed to be lognormally distributed and geometric average of a set of lognormally distributed variables is also lognormal). The geometric average is defined as

$$G_T = \left(\prod_{j=1}^m S_j \right)^{1/m}, \quad m = T / \Delta t.$$

Therefore the price of the geometric Asian call option is given by a modified Black-Scholes formula [Clewlow at al.]:

$$(35) \quad C_{G_Asian} = e^{-rT} \cdot (\exp(a + b/2) \cdot N(x) - K \cdot N(x - \sqrt{b})),$$

where

$$a = \ln(G_t) + \frac{m-k}{m} \left[\ln(S_0) + \left(\mu - \frac{\sigma^2}{2} \right) (t_{k+1} - t) + \frac{1}{2} \cdot \left(\mu - \frac{\sigma^2}{2} \right) (T - t_{k+1}) \right],$$

$$b = \left(\frac{m-k}{m} \right)^2 \cdot \sigma^2(t_{k+1} - t) + \frac{\sigma^2(T - t_{k+1})}{6m^2} (m-k)(2(m-k)-1), \quad x = \frac{a - \ln(K) + b}{\sqrt{b}},$$

where G_t is the current geometric average and k is the last known fixing.

A common approach to pricing an arithmetically averaged Asian option is to use the geometrically averaged Asian option as the control variate (Chapter 3.2.2).

The Monte Carlo (QMC) implementation to price this derivative:

1. For each i -th path we find $S_{ave}^{(i)} = A_T = \frac{1}{m} \sum_{j=1}^m S_j^{(i)}$ and

$$G_T^{(i)} = \left(\prod_{j=1}^m S_j^{(i)} \right)^{1/m}, \quad m = T / \Delta t, \quad i = \overline{1, N}.$$

2. Calculate payoff of the i -th path. For example, for an average price call the payoff is $C_{Portf}^{(i)} = \max(0, S_{ave}^{(i)} - K) - \max(0, G_T^{(i)} - K)$, $i = \overline{1, N}$. Here we take $\beta = 1$ for simplicity.
3. Discount this value by the risk-free rate to get the price of the option: $\hat{C}_{portf}^{(i)} = e^{-rT} \cdot C_{portf}^{(i)}$.

4. Take an average of those payoffs: $\hat{C}_{Portf} = \frac{1}{N} \cdot \sum_{i=1}^N \hat{C}_{Portf}^{(i)}$.

5. Compute the standard error: $SE(\hat{C}_{ave}) = \frac{\sqrt{\frac{1}{N-1} \sum_{i=1}^N (\hat{C}_{Portf}^{(i)} - \hat{C}_{Portf})^2}}{\sqrt{N}}$.

6. Calculate the price of the Asian call option with control variate as $C_{Asian_c} = \hat{C}_{Portf} + C_{G_Asian}$, where price of the geometrical Asian call option can be found by formulas (35).

Applying the Monte Carlo technique to more complicated options such as path-dependent options requires a partitioning of the option's life into time periods, as in the binomial model, so the longer life of the option the higher dimension problem we have and the more computational time is needed to price this option. For example, to price an Asian option, in which the average price be computed by collecting the daily closing price over 100 day life. Then a run would consist of 100 random drawings, each used to simulate the stock price at the end of each of the 100 days. The formula for each S would be based on the previous day's closing price. The value of Δt would be $1/365$. Then the average of the 100 stock prices would determine the option payoff at expiration. You would then need to repeat the procedure at least 50,000 times. The more complex options, however, would probably require at least 100,000 runs.

By applying both antithetic variates and control variates techniques to value the Asian call option we can significantly reduce the standard error (by 37 times) whereas the computation time increases by 30%.

Suppose we want to evaluate an Asian option with following parameters: $S_0 = 100$, $K = 100$, $T = 1$, $r = 5\%$, $\delta = 2\%$, $\sigma = 0.10$.

Table below represent the results of evaluating the Asian call option by different variance-reduction techniques. Sobol sequences were used to generate uniform samples in QMC method.

Table 13. Comparison of different variance reduction techniques for Asian Call ($N=1000$, $s=12$)

	Price	Standard Error	Computation time (sec)
<i>Simple MC</i>	3.3182	0.1348	7
<i>Simple QMC (none of VR techniques used)</i>	3.1675	0.1295	6
<i>Antithetic variate</i>	2.9209	0.1161	9
<i>Control variate*</i>	2.9887	0.0026	6
<i>Latin Hypercube</i>	3.3236	0.1361	6
<i>Brownian Bridge</i>	3.1778	0.1291	7

* Geometric mean Asian option is used as Control Variate

Using QMC method we could decrease the standard error comparing to simple Monte Carlo, and this difference rises with increasing number of paths. So, in this case the implementation of antithetic variates allows us to decrease the standard error but the computation time increases. Also the significant variance reduction can be obtained by using geometrical mean option as a control variate. Control variate reduces the standard error by approximately 52 times. To achieve this reduction with the simple Monte Carlo would require increasing the number of simulations by $52 \times 52 = 2,704$ times that would result in significant increase in the computation time.

4.4 Evaluating Mortgage with Prepayment Options

Mortgage-backed securities [Fabozzi] are securities backed by a pool (collection) of mortgage loans. Mortgage-backed securities subdivide into 1) mortgage passthrough securities; 2) collateralized mortgage obligations; and 3) stripped mortgage-backed securities. A *mortgage loan* (mortgage) is a loan that secured by the collateral of some specified real estate property which obliges the borrower to make a predetermined series of payments. The interest rate on the mortgage loan is called *the mortgage rate* or *contract rate*.

The cash flows of a mortgage backed-security consist of net interest (interest after servicing and guarantor fees), regularly scheduled principal payments, and prepayments. In order to price a mortgage-backed security, all its cash flows need to be projected. But usually cash flows are unknown because of prepayments. The only way to project cash flows is to make some assumption about the prepayment rate over the life of the underlying mortgage pool. The prepayment rate i sometimes referred to as the *speed*. Conditional prepayment rate and Public Securities Association prepayment benchmark are two conventions that could be used as a benchmark for prepayment rates. Prepayment rates are affected by following factors: 1) the prevailing mortgage rate, 2) characteristics of the underlying mortgage, 3) seasonal factors, and 4) general economic activity.

The periodic cash flows of a mortgage-back security are *interest rate path-dependent* due to 1) refinancing burnout, and 2) payments rules for collateralized mortgage obligations (CMO). A *path-dependent cash flow* is one in which the cash flow received in one period depends not only on the current interest rate level, but also on the path that interests took to get the current level. Monte Carlo (QMC) simulation (Chapter 2) is the most flexible and effective method to value these interest rate path-dependent fixed income securities. The Monte Carlo model involves randomly generating many scenarios of future interest rate paths. The interest rate paths are generated based on some volatility assumption for interest rates. In case of mortgage backed securities, this model involves generating a set of cash flows based on simulated future mortgage refinancing rates. The random paths of interest rates should be generated from an arbitrage-free model of the future term structure of interest rates. An arbitrage-free model is a model that replicates today's term structure of interest rates, an input of the model, and for all future dates there is no possible arbitrage within the model.

Assume we want to simulate N paths for M -dimensional security (for example, a mortgage loan that matures in $M=360$ months with monthly payments). To value a mortgage-backed security using Monte Carlo (QMC) technique we can follow the steps

1. For each path ($i = \overline{1, N}$) simulate M monthly future interest rates $f_j(i)$, $j = \overline{1, M}$. For example, $f_j(i)$ is a simulated future 1-month interest rate for month j on path i . These could be done by techniques discussed in Chapter 3.1.4 implementing MC or QMC for generating uniform random numbers.
2. Simulate paths of monthly spot rates $z_j(i)$, $j = \overline{1, M}$, $i = \overline{1, N}$. The spot rate for month j on a path i can be obtained from the simulated future rates along the path:

$$z_j(i) = \left\{ [1 + f_1(i)] \cdot [1 + f_2(i)] \cdot \dots \cdot [1 + f_j(i)] \right\}^{1/j} - 1, \quad j = \overline{1, M}, \quad i = \overline{1, N}.$$
3. For each path ($i = \overline{1, N}$) simulate M monthly mortgage refinancing rates $r_j(i)$, $j = \overline{1, M}$. They are needed to determine the cash flows because it represents the opportunity cost the mortgagor is facing at that time. For example, if the refinancing rates are high relative to the original mortgage rate, the mortgagor will have less incentive to refinance and opposite.
4. Simulate Cash flows $C_j(i)$, $j = \overline{1, M}$ on each of the interest rate paths $i = \overline{1, N}$, based on simulated mortgage refinancing rates $r_j(i)$, $j = \overline{1, M}$, $i = \overline{1, N}$.
5. Calculate the present value of the cash flows for month j , $j = \overline{1, M}$, on i -th path as follow (K is spread):

$$PV[C_j(i)] = \frac{C_j(i)}{[1 + z_j(i) + K]}, \quad j = \overline{1, M}, \quad i = \overline{1, N}.$$

6. Compute the present value for each path $PV[Path(i)]$, $i = \overline{1, N}$, as the sum of the present value of the cash flow for each month on this path:

$$PV[Path(i)] = PV[C_1(i)] + PV[C_2(i)] + \dots + PV[C_M(i)] = \sum_{j=1}^M PV[C_j(i)], i = \overline{1, N}$$

7. Obtain the theoretical value of the security by calculating the average of the theoretical values of all the interest rate paths (N is number of interest rate paths).

$$\text{Theoretical value} = \text{Price} = \frac{\sum_{i=1}^N PV[Path(i)]}{N}.$$

The theoretical value of a security on any interest rate path is the present value of the cash flows on that path where the spot rates are those on the corresponding interest rate path.

Using the information above we can also simulate the average life of a mortgage-backed security as the weighted average time to receipt of principal payments (scheduled payments and projected prepayments). The average life of the security along the N paths is

$$\text{Average life} = \frac{1}{N} \sum_{i=1}^N \text{Av.Life}(i),$$

where $\text{Av.Life}(i)$ is the average life of the security along the path $i, i = \overline{1, N}$, and

$$\text{Av.Life}(i) = \frac{\sum_{j=1}^M j \cdot (\text{Principal at time } j)}{12 \cdot (\text{Total Principle received})}, M \text{ is number of months.}$$

To provide reliable estimate of the price of a mortgage-backed security, some adjustment to the interest rate paths should be made to prevent interest rates from reaching levels that are believed to be unreasonable (e.g. an interest rate of zero or below or an interest rate too high (more than 30%). This can be done by mean reversion: the interest rate is forced toward some estimated average (mean) value. Also, the correlation between the short-term rates and refinancing rates must be estimated.

Some form of variance-reduction techniques (Chapter 3.2-3.3) could be implemented to reduce the number of sample paths needed to achieve a good statistical sample. With variance reduction methods we can obtain price estimates within a tick. That means that generating more scenarios than N will not cause the price estimates to change more than a tick.

As an example, consider a mortgage of \$100,000 for $T=1$ and $T=3$ years payable monthly. Current annual interest rate is 5%. Volatility is 4%.

Fig. 20 Brownian motion path for the term structure ($M=12$)

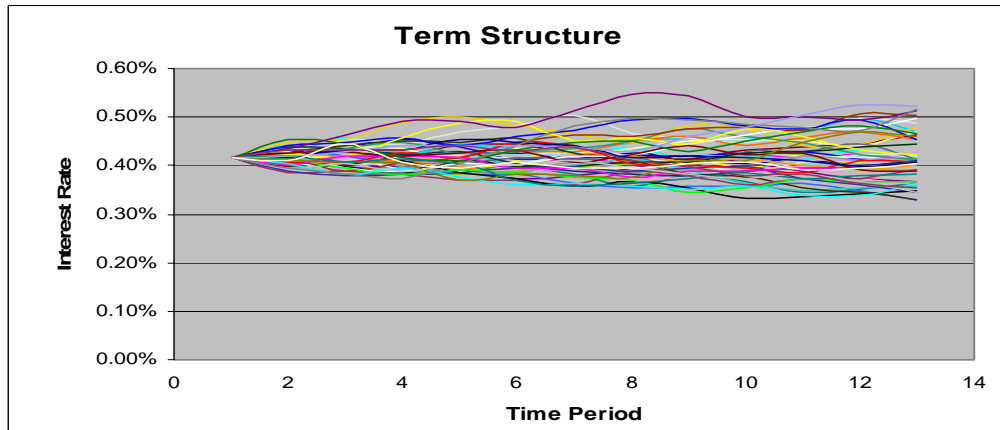


Fig. 21 Brownian motion path for the term structure ($M=36$)

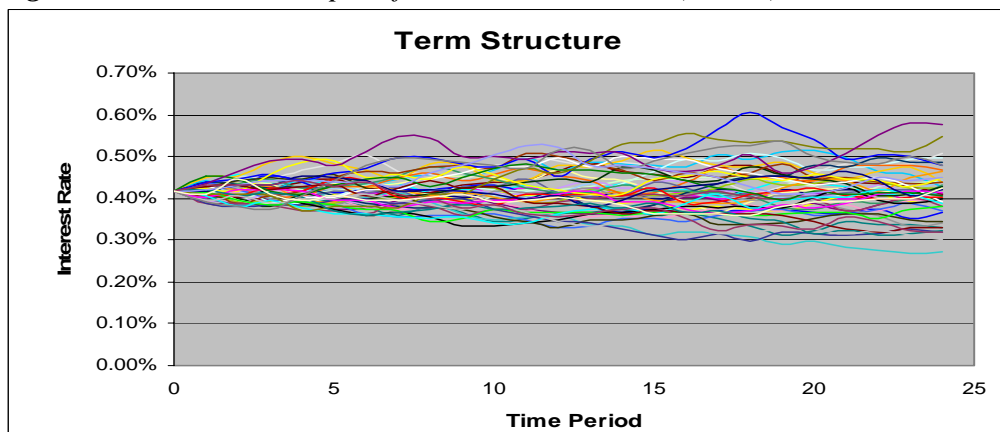


Table 14. Comparison of MC and QMC methods for pricing Mortgage-Backed security ($N=1000$, $s=12$) with prepayment option

	M=12	M=12	M=36	M=36
	Standard Error (w/o prepayment)	Standard Error (with prepayment)	Standard Error (w/o prepayment)	Standard Error (with prepayment)
<i>Simple MC</i>	5.86	4.00	26.42	3.69
<i>QMC</i>	5.54	3.87	16.52	3.03

As we can notice again quasi-random sequences perform better in this problem too. To generate quasi-random numbers Faure sequences were used. With increasing of dimension the standard error without prepayment in case of both MC and QMC methods increases also but the percentage of increase for QMC methods is smaller than in case of simple Monte Carlo (the standard error increases by 2.98 times for QMC comparing to 4.5 times for MC). The standard error with prepayment in case of both MC and QMC methods decreases but once again the percentage of decrease for QMC methods is higher than in case of simple Monte Carlo (the standard error decreases by 1.28 times for QMC comparing to 1.08 times for MC). The results confirm our previous conclusions about comparison between MC and QMC methods. Using the same variance-reduction techniques simultaneously for both MC and QMC methods will result in

even more distinction in accuracy. Also, the difference in accuracy become larger when larger number of sample is generated to price this security.

CONCLUSIONS

In this paper first we gave a review of Monte Carlo and quasi-Monte Carlo with stating advantages and disadvantages of each method and independent comparison pseudo-random and quasi-random sequences.

Some summary of the finding in this paper regarding MC and QMC methods:

- Low discrepancy sequences sets routinely outperformed the pseudo-random sequences. Quasi-random sequence presents a better performance than typical pseudo-random sequences for all four probabilistic moments, indicating that quasi-random sequence is more representative of $U[0, 1]$ than pseudo-random numbers.
- The Sobol sequence can be generated significantly faster than the Faure sequence and faster than most pseudo-random number methods. Advantages the Sobol sequence over Faure and Halton sequences is due to the use base 2 for all dimensions. So, there is some computational time advantage due the shorter cycle length. The time to construct Hybrid sequences is larger than for Halton sequences due to additional operations needed to calculate uniform permutations of the one-dimension sequence (base 2). Faure sequences consume much more times that all other tested here sequences. As an advantage of pseudo-random sequences is that the point sets can be reused if they are shifted.
- The Sobol sequence gives the least standard error, comparing to other methods. All quasi-random sequences perform better than pseudo-random, but Hybrid method gives the worst error among other low-discrepancy sequences. Faure sequences are outperforms all other sequences but Sobol ones, but we should remember from previous comments that Faure sequences consume more time.
- Quasi-random methods, especially Sobol sequence, can achieve the given accuracy with much less samples (sometimes in 10-1000 times less) than pseudorandom sampling requires. Moreover, larger number of samples used by simple Monte Carlo to achieve this accuracy will result in significant increase of the computational time.
- To improve accuracy one would require more pseudo-random numbers than low-discrepancy ones.
- The Halton sequence is dominated by the Faure and Sobol sequences. Although an implementation of Halton method is much easier than Sobol and Faure methods, it is not wise to use them in high dimensions. For example, in some finance application its performance falls dramatically with increasing dimensions.
- The Monte Carlo method is sensitive to the initial seed.
- For functions that typically arise in security pricing, many studies have found the performance of QMC methods to equal or exceed that of standard Monte Carlo.
- Hybrid Quasi method shows no degradation in high dimensions, but its uniform properties are worse than of other LD sequences.
- In low dimensions, the performance of some low discrepancy sequences is often much better than Monte Carlo.
- With higher dimensions problem, it might be good to use Sobol sequences to be positive in fast convergence and reliability.
- Some problems with Low Discrepancy sequences are: 1) high-dimensional clustering; possible high correlation of neighboring dimensions; 3) possible transformation difficulties.

- The major problem for the simple quasi-random sequences is high-dimensional clustering. According our analysis Halton and the Faure sequences suffer from this problem with much lower dimension. Hybrid sequences as well as pseudo-random sequences do not have such problem, but as we noted before, they result in lower standard error and could not be considered as a good alternative to low-discrepancy sequences. The Sobol sequence appears to resist more to the high-dimensional degradation. One of the solutions to prevent high-dimensional clustering near zero is to discard first n points. Faure sequences perform better if first $n=Prime\ number\ used\ to\ generate\ this\ sequence$ are discard.
- Some quasi-random sequences in high-dimensional problems are highly correlated. Halton sequence becomes unsatisfactory after dimension 14. Because the Hybrid quasi-points built as permutations of Corput sequence with base 2, this method does not experience problems with autocorrelation in high-dimension environment. The Faure sequence exhibits high-dimensional degradation at approximately the 25th dimension. Sobol sequences also show some correlation between neighboring dimensions, but this correlation is lower than for other low-discrepancy sequences and started at higher dimension.
- To implement QMC methods to finance problems, generated quasi-random sequences must have not good only uniform quality but also posses a good normal quality after using some transformation techniques. Chapter 2.5 demonstrates usefulness of Sobol sequences to get a good practical implementation of normality.
- The relative advantage of QMC over standard MC decreases as the dimension of the problem increases.
- For some high-dimensional problems the relative advantage of QMC can be increased by judicious use of the sequences.
- Sometimes problems with a high nominal dimension may have a much lower effective dimension (with uses of Brownian Bridge or principal component constructions).

As was shown in Chapter 4 the performance of quasi-Monte Carlo methods can also be improved by using traditional variance reduction techniques (importance sampling, stratification, antithetic variables, control variables, Brownian Bridge, Latin hypercube, or even implementing several variance-reduction techniques simultaneously). Here we need to be aware of possible increasing computational time when dealing with antithetic variates. Before implementing specific variance-reduction techniques, following recommendation might be helpful.

Antithetic sampling effective if

- $Cov(W_t, \tilde{W}_t) < 0$, this way the values computed from a path and its mirror image are negatively correlated.
- f is monotone in each of argument. Particular, if f is linear then the antithetic estimate of $E[f(Z_1, \dots, Z_m)]$ has zero error.
- f is symmetric. In this case the variance of an antithetic sample of size $2m$ is the same as an independent sample of size m .

The control variates technique is efficient when

- There are some easily computed quantities that are highly correlated with the object of interest.
- This method is especially good for pricing Asian options depending on geometric averaging. In this case we can take the geometric average as a control variate to price Asian options which use arithmetic averaging.
- The control variate could also be successfully used when we deal with two similar options: a simpler option problem which has an analytic solution, and uses that solution to improve the accuracy of the more complex problem at hand (that has no analytic solution).

Importance sampling is

- Can be efficiently used to make rare events less rare.
- Efficient with eliminating singularities in the objective function, or with reduction of a problem with infinite variance to one with finite one.
- one needs to be very careful with choosing a good sampling density.

Latin hypercube

- Asymptotically eliminates the contribution to the variance due to the additive part of the function being integrated (that is advantage of this method).
- Does not provide a means of estimating a standard error to be used to form a confidence interval around a point estimate (that is disadvantage of this method).
- Technique requires careful batching of the sample data because the sample points are highly correlated in hypercube sampling. To obtain a good estimate of the error we should relax some of the variance reduction properties.

Brownian Bridge

- Can result in decreasing of effective dimension, because by the construction more importance is given to the first few uniform numbers (advantage).
- Can demand additional computation complexity/time (reordering the path) for path-dependent problems (disadvantage).

According to Chapter 3.3, Brownian Bridge Discretization for some financial problems provides a good way to reduce variance.

To summarize this work, next steps should be done to evaluate path-dependent or other financial security using Monte Carlo or QMC.

1. Generate a matrix of uniform random numbers. Pseudo-random or quasi-random numbers discussed in Chapter 2.3-2.4 could be used.
2. Transform this matrix of uniform random numbers to a matrix of the same size with standard normal numbers. Moro's inversion method discussed in Chapter 2.5 could be used.
3. Generate Brownian motion paths based on the matrix of standard normal numbers. Corresponding methods from Chapter 3.1 could be used.
4. Evaluate the given financial security using information along the Brownian motion paths. Chapter 4 gives examples of evaluating European options, Asian options, Barrier options, and mortgage-backed securities with prepayment option.
5. To achieve higher accuracy some variance-reduction techniques could be implemented. Such techniques as were discussed in Chapter 3.2 could be used. One should carefully investigate the given financial problem to understand which variance-reduction technique would be more appropriate and effective in that case. As was shown in Chapter 4, for different financial problems different variance-reduction techniques could result in most variance reduction. Thus, for example, with Geometrical mean Asian option as Control variate could be achieved a significant decrease in error for Asian option, while Delta as Control variate for European option is not as good as addition Latin hypercube sampling and Brownian Bridge construction.

BIBLIOGRAPHY

1. Acworth P., Broadie M., Glasserman P. A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing. In P. Hellekalek and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 127 in Lecture Notes in Statistics, pp. 1-18, New York, 1997. Springer-Verlag.
2. Akesson F., Lehoczy J.P. Path generation for quasi-Monte Carlo simulation of mortgage-backed securities. *Management Science*, 46:1171, 1187, 2000.
3. Antonov I.A., Saleev V.M. An economic method of computing LP_r -sequences. *Zh. Vychisl. Mat. I Mat. Fiz.*, 19, 1997, pp. 243-245 (In Russian).
4. Barret, J.W., Moore G., Wilmot P. Inelegent Efficiency. *Risk*, vol.5, n° 9, 1992, pp.82-84.
5. Beasley J.D., Springer S.G. The Percentage Points of the Normal Distribution, *Applied Statistics* **23**, 1997, pp.118-121.
6. Black F., Scholes M. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637 654, 1973.
7. Bratley P., Fox B.L, and Niederreiter H. Implementation and tests of low-discrepancy sequences. *ACM Transactions on Mathematical Software*, 14, pp.88-100, 1992.
8. Broadie M., Glasserman P. *Risk Management and Analysis. Vol.1.* 1998. John Wiley & Sons Ltd.
9. Broadie M., Glasserman P., Kou S.G. A Continuity Correction for Discrete Barrier Options. *Mathematical Finance*, **7**, № 4, 1997, pp.325-349.
10. Boyle P. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4:323 338, 1977
11. Boyle P., Broadie M., Glasserman P. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, **21**, 1997, pp. 1267-1321.
12. Caflisch R.E., Moskowitz B. Modified Monte Carlo methods using quasi-random sequences. In H. Niederreiter and P.J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 106 in Lecture Notes in Statistics, pp. 1-16, New York, 1995. Springer-Verlag.
13. Clewlow L., Strickland C. *Implementing Derivatives Models.* John Wiley & Sons Ltd., 1998, p.308.
14. Cox J., Ingersoll J.E., Ross S.A. A theory of the term structure of interest rates. *Econometrica*, **53**, 1985, pp. 385-407.
15. Fabozzi F. *Valuation of Fixed Income Securities and Derivatives.* Third Edition.
16. Faure H. Discrepance de suites associees a un systeme de numeration (en dimension s). *Acta Arithmetica*, **41**, 1982, pp.337-351.
17. Galanti S., Jung A. Low-Discrepancy Sequences: Monte Carlo Simulation of Option Prices. *Journal of Derivatives*. Fall 1997, pp.63-83.
18. Heath D., Jarrow R., Morton A. Bond Pricing and the term structure of interest rates: A new methodology for contingent claims valuation. *Econometrica*, **60**, 1992, pp.77-105.
19. Hull J.C. *Options, Futures, and Other Derivatives.* Fifth Edition. Prentice Hall, 2003, pp.740.
20. Hull J., White A. The pricing of options on assets with stochastic volatilities. *Journal of Finance*, **42**, pp. 281-300.
21. Jaeckel P. *Monte Carlo Methods in Finance.* Wiley, New York, 2002.
22. Joy C., Boyle P.P., Tan K.S. Quasi-Monte Carlo Methods in Numerical Finance. *Methodologies and Applications for Pricing and Risk Management*, Number 24, pp. 269-280.

23. Jung A. Improving the Performance of Low-Discrepancy Sequences. *Journal of Derivatives*. Winter 1998, pp.85-95.
24. Lemieux C., L'Ecuyer P. *On the Use of Quasi-Monte Carlo Methods in Computational Finance*. Department of Mathematics and Statistics, University of Calgary, Canada.
25. McKay M.D., Conover W.J., Beckman R.J. A comparison of three methods for selecting input variables in the analysis of output from a computer code. *Technometrics*, **21**, 1979, pp.239-245.
26. Moro B. The Full Monte. *Risk*, **8**, № 2, February 1995.
27. Morokoff W.J., Caflisch R.E. Quasi-Monte Carlo simulation of random walks in finance. In P. Hellekalek and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 127 in Lecture Notes in Statistics, pp. 340-352, New York, 1997. Springer-Verlag.
28. Niederreiter H. *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF, Vol. 63. Philadelphia: SIAM, 1992.
29. Ninomiya S., Tezuka S. Toward real-time pricing of complex financial derivatives. *Applied Mathematical Finance*, **3**, pp 1-20, 1996.
30. Owen, A.B. Monte Carlo Extension of Quasi-Monte Carlo. *Working Paper*, Stanford University, 2000, 7 pp.
31. Paskov S. New Methodologies for Valuing Derivatives. *Monte Carlo: Methodologies and Applications for Pricing and Risk Management*, number 25, pp.281-298.
32. Paskov S., Traub J. Faster valuation of financial derivatives. *Journal of Portfolio Management*, 22:113 120, 1995
33. Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. Quasi- (that is, Sub-) Random Sequences. §7.7 in *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd ed. Cambridge, England: Cambridge University Press, pp. 299-306, 1992.
34. Silva M.E. Quasi Monte Carlo in Finance: Extending for High Dimensional Problems. *Resenha da BM&F*, 151, 2002
35. Sobol I.M. On the distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Computational Mathematics and Mathematics Physics*, **7**, pp.86-112, 1967.
36. Ripley, B.D. *Stochastic Simulation*. John Wiley & Sons, Inc., 1987, 237 p.
37. Tezuka S. Financial Applications of Monte Carlo and Quasi-Monte Carlo Methods. *Random and Quasi-Random Point Sets*, P. Hellekalek & G. Larcher, Eds., Springer-Verlag New York, 1998, pp.303-332.
38. Willard G.A. Calculating Prices and Sensitivities for Path-Independent Derivative Securities in Multifactor Models. *Journal of Derivatives*, Fall 1997, pp.45-61.