FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

PROBABILISTIC METHODS IN ESTIMATION AND PREDICTION OF FINANCIAL MODELS

By

NGUYET THI NGUYEN

A Dissertation submitted to the
Department of Mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Summer Semester, 2014

Nguyet Thi Nguyen defended this dissertation on July 10, 2014.
The members of the supervisory committee were:

Giray Ökten
Professor Directing Dissertation

Lois Hawkes
University Representative

Bettye Anne Case
Committee Member

Kyounghee Kim
Committee Member

Warren Nichols
Committee Member

Jinfeng Zhang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

I dedicate this dissertation to my family.

ACKNOWLEDGMENTS

First of all, I sincerely thank my advisor, professor Giray Ökten. His endless effort to support me is absolutely crucial for me to complete this dissertation. He is not only a great research director, but also an amazing career mentor. I am lucky to work with him for the last four years and what I have learned from him will definitely be very helpful for my future career.

I would also like to thank professors: Lois Hawkes, Bettye Anne Case, Jinfeng Zhang, Warren Nichols, and Kyounghee Kim, for serving on my committee and assisting me in any way I needed.

During my studying time at Florida State University, I received valuable encouragement and help from many professors and friends. I would like to thank professor Case for encouraging and supporting me since I just came to the mathematics department. I wish to thank faculty members: Warren Nichols, David Kopriva, Penelope Kirby, and Annette Blackwelder they have supported me and given me the inspiration to be a good teacher. I thank Dr. John Burkardt, Department of Scientific Computing, for the inverse transformation codes, and my classmate Linlin Xu, for his fast random start Halton codes used in this dissertation. I also would like to thank all of my friends who gave me their hands whenever I needed help.

Last, but not least, I would like to thank my whole family. Without their support, it would have been impossible for me to finish my dissertation. The completion of my study at Florida State University is an achievement of all my family members: my husband Dr. Huan Tran; my daughters Minh Tran and Nina Tran; my parents Binh Nguyen and Chi Hoang; my parents in law Dr. Bien Tran and Yen Mai; and my brother Huyen Tran.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Many computational finance problems can be classified into two categories: estimation and prediction. In estimation, one starts with a probability model and expresses the quantity of interest as an expected value or a probability of an event. These quantities are then computed either exactly, or numerically using methods such as numerical PDEs or Monte Carlo simulation. Many problems in derivative pricing and risk management are in this category. In prediction, the main objective is to use methods such as machine learning, neural networks, or Markov chain models, to build a model, train it using historical data, and predict future behavior of some financial indicators.

In this dissertation, we consider an estimation method known as the (randomized) quasi-Monte Carlo method. We introduce an acceptance-rejection algorithm for the quasi-Monte Carlo method, which substantially increases the scope of applications where the method can be used efficiently. We prove a convergence result, and discuss examples from applied statistics and derivative pricing. In the second part of the dissertation, we present prediction algorithms based on hidden Markov models. We use the algorithms to predict and evaluate economic regimes, and stock prices, based on historical data.

CHAPTER 1

INTRODUCTION

1.1 Monte Carlo methods

The Monte Carlo simulation is a popular numerical method across sciences, engineering, statistics, and computational mathematics. In simple terms, the method involves solving a problem by simulating the underlying model using pseudorandom numbers, and then estimates the quantity of interest as a result of the simulation. In recent years, a deterministic version of the MC method, the so-called quasi-Monte Carlo (QMC) method, has been widely used by researchers. Different from the MC method which relies upon pseudorandom numbers, QMC method uses low-discrepancy sequences for the sampling procedure. The QMC method converges generally faster than the corresponding MC method. In this section, we review some basic definitions and theorems related to these methods which are necessary for our work presented in the subsequent chapters.

It is convenient to describe the MC and QMC method in the context of evaluating a multi-dimensional integral

$$Q = \int_{I^s} f(\mathbf{x}) d\mathbf{x}. \tag{1.1}$$

Here $I = [0, 1]$ and s a positive integer. In the MC method, we generate N pseudorandom vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ in I^s, and estimate the integral (1.1) by

$$\overline{Q}_N = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i). \tag{1.2}$$

The strong law of large numbers states that

$$\lim_{N \to \infty} \overline{Q}_N = Q. \tag{1.3}$$

The performance of the estimator $\overline{Q}_N$ is examined in terms of the error, defined as

$$E_{\text{MC}} \equiv Q - \overline{Q}_N. \tag{1.4}$$

Now suppose that f has finite variance

$$\sigma^2(f) = \int_{I^s} [f(\mathbf{x}) - Q]^2 d\mathbf{x} < \infty. \tag{1.5}$$

By the central limit theorem, we have

$$E_{\mathrm{MC}} \approx \mathcal{N}\left(0, \frac{\sigma^2(f)}{N}\right), \tag{1.6}$$

where $\mathcal{N}(0, \sigma^2(f)/N)$ is the normal distribution with mean zero and variance $\sigma^2(f)/N$. Therefore the error bound of the MC method is $O(N^{-1/2})$. The MC method has many advantages:

- It is simple and easy to implement on a computer. It does not require specific knowledge of the form of the solution or its analytic properties.
- The error bound is independent of the dimension thus the MC method offers a way of overcoming the curse of dimensionality.
- In general, it is easy to parallelize a MC algorithm. Multiple processors can run a Monte Carlo simulation simultaneously since each simulation is independent of another.

However, the MC method for numerical integration has the primary drawback:

- The convergence of MC method is slow. Since the probabilistic error bound of MC method is $E_{MC} \approx O(N^{-1/2})$ many samples may be required to obtain acceptable precision in the answer. In particular, to achieve one more decimal digit of precision in the answer requires increasing the sample size by a factor of 100.

To improve the efficiency of the MC method for numerical integration, a number of techniques have been developed. One of the techniques is using low-discrepancy sequences instead of the pseudorandom sequences used by the MC method. We will discuss these sequences in the next section.

1.2 Quasi-Monte Carlo methods

The QMC method is based on the theory of uniform distribution and discrepancy. In this section we will give a brief review of some of the definitions and theorems. The proofs of the theorems can be found in Niederreiter [46].

1.2.1 Uniform distribution modulo 1

We begin this section with the definition of a counting function. Let $\omega = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$ an infinite sequence of real numbers in I^s, and $\omega_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. For an arbitrary subset E of I^s, we define

$$A(E; \omega_N) = \sum_{n=1}^{N} c_E(\mathbf{x}_n), \tag{1.7}$$

where c_E is the characteristic function of E.

Definition 1 *A sequence ω is said to be uniformly distributed modulo 1 (u.d mod 1) if for an arbitrary subset E of I^s, we have*

$$\lim_{N\to\infty} \frac{A(E;\omega_N)}{N} = \lambda_s(E), \tag{1.8}$$

where λ_s is the Lebesgue-measure on I^s.

Note that equation (1.8) is equivalent to

$$\lim_{N\to\infty} \sum_{n=1}^{N} \frac{1}{N} c_E(\mathbf{x}_n) = \int_{I^s} c_E(\mathbf{x}) d\mathbf{x}. \tag{1.9}$$

Theorem 1 *A sequence ω is u.d. mod 1 if and only if for every real-valued continuous function f defined on I^s we have*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{x}_n) = \int_{I^s} f(x) dx. \tag{1.10}$$

Corollary 1 *A sequence ω is u.d. mod 1 if and only if for every Riemann-integrable function f on I^s the equation (1.10) holds.*

The convergence of the function $A(E,\omega_N)/N$ depends not only on the characteristics of the sequence ω, but also on properties of the set E. We will define a class of sets E on which Equation (1.8) holds for any u.d. mod 1 sequence.

Definition 2 *A Borel set $E \subseteq I^s$ is called a $\lambda_s -$ continuity set if $\lambda_s(\partial E) = 0$,*

where ∂E is the boundary of the set E.

Theorem 2 *If ω is u.d. mod 1 then*

$$\lim_{N\to\infty} \frac{A(E;\omega_N)}{N} = \lambda_s(E) \tag{1.11}$$

for all λ_s-continuity sets E in I^s.

1.2.2 Discrepancy

The uniformity of the sequence ω can be quantified by its discrepancy, which is defined next.

Definition 3 *Let $\omega_N = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ be a finite sequence of real numbers in I^s, and $\mathcal{J}$ be a nonempty family of Lebesgue-measurable subsets of I^s. The number*

$$D_N(\omega_N) = D_N(\mathcal{J}, \omega_N) = \sup_{J \in \mathcal{J}} \left| \frac{A(J; \omega_N)}{N} - \lambda_s(J) \right| \tag{1.12}$$

is called the discrepancy of ω_N.

For any vector $\mathbf{x} = (x_1, x_2, ..., x_N)$ in I^s, let $\mathcal{J}^*$ be a family of subintervals $[0, \mathbf{x}) = \prod_{i=1}^{s} [0, x_i)$ of I^s. The discrepancy $D_N^*(\omega_N) = D_N(\mathcal{J}^*, \omega_N)$ is called the star-discrepancy of ω_N. The relationship between a u.d. mod 1 sequence with its discrepancy is expressed by the following theorem.

Theorem 3 *The sequence ω is u.d. mod 1 if and only if*

$$\lim_{N \to \infty} D_N(\omega_N) = 0.$$

Theorem 4 *For any sequence ω_N in I, we have*

$$\frac{1}{N} \leq D_N(\omega_N) \leq 1. \tag{1.13}$$

From Theorem 4, we see that the discrepancy of a sequence has a fixed upper bound of one and a lower bound that approaches zero as N goes to infinity. The discrepancy is also bounded by its star-discrepancy.

Theorem 5 *For any point set $\omega_N \in I^s$, we have*

$$D_N^*(\omega_N) \leq D_N(\omega_N) \leq 2^s D_N^*(\omega_N). \tag{1.14}$$

From Theorems (3) and (5) we have the following corollary.

Corollary 2 *The sequence ω is u.d. mod 1 if and only if*

$$\lim_{N \to \infty} D_N^*(\omega_N) = 0.$$

1.2.3 Error bounds for QMC

The classical QMC error bound is the celebrated Koksma-Hlawka inequality [46] given in the following theorem.

Theorem 6 *(Koksma-Hlawka inequality) If f is a function of bounded variation, with the variation $V(f)$, on $\overline{I}^s$, then for any low-discrepancy point set $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ in I^s, we have*

$$E_{QMC} = \left| \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i) - \int_{I^s} f(\boldsymbol{x}) d\boldsymbol{x} \right| \leq V(f) D_N^*(\omega). \tag{1.15}$$

The inequality is based on the assumption that the integrand is a bounded variation function in the sense of Hardy and Krause (BVHK). The accuracy of the estimation is based on the star-discrepancy of the point set ω, and the variation $V(f)$. The star-discrepancy of the low-discrepancy sequences is of $O(\ln^s(N)/N)$. Therefore, the QMC method has the error bound $O(\ln^s(N)/N)$ for an integral of a BVHK function. This rate is asymptotically faster than the MC rate $O(N^{-1/2})$.

1.2.4 Examples of low-discrepancy sequences

Van der Corput and Halton sequences. Let $b \geq 2$ be an integer. The van der Corput sequence $\{\phi_b(n)\}_{n=0}^{\infty}$ in base b is defined as follows:
For any positive integer n, suppose that the expression of n in base b is

$$n = (a_m ... a_1 a_0)_b = \sum_{j=0}^{m} a_j b^j,$$

then

$$\phi_b(n) = \sum_{j=0}^{m} \frac{a_j}{b^{j+1}}.$$

The function $\phi_b(n)$ is called the *radical-inverse* function. Normally, we choose b as a prime number. The following upper bound for the discrepancy of the van der Corput sequence in base $b = 2$ is from Niederreiter and Kuipers [31].

Theorem 7 *The discrepancy $D_N(\phi_2(n))$ of the van der Corput sequence $\phi_2(n)$ satisfies*

$$D_N(\phi_2(n)) \leq \frac{\ln(N+1)}{N \ln 2}. \tag{1.16}$$

Introduced in 1960 by Halton [28], the Halton sequence is a generalization of the van der Corput sequence to higher dimensions. The s-dimensional Halton sequence in the bases $b_1, ..., b_s$ is defined as $\omega = \{(\phi_{b_1}(n), ..., \phi_{b_s}(n)), n = 1, 2, ...\}$. The Halton sequence is u.d. mod 1 if its bases $b_1, ..., b_s$ are relatively prime. In practice, we choose b_k as the k^{th} prime number. Meijer [45] gave the following upper bound for the star-discrepancy of the s-dimensional Halton sequence:

$$D_N^*(\omega_N) \leq \prod_{j=1}^{s} \frac{b_j - 1}{\ln b_j} (\log N)^s + O((\log N)^{s-1}). \tag{1.17}$$

This bound has been improved significantly since then in [1] and [21].

Faure sequences. The Faure sequence was introduced by Faure in 1982 [20]. Like the Halton sequence, the Faure sequence is an extension of the van der Corput sequence, however it uses only one base for all dimensions and uses a permutation of the vector elements for each dimension. The base for all dimensions is the smallest prime b which is bigger than or equal to the dimension s.

Joy, Boyle and Tan [30] described how to generate the n^{th} element of the s dimensional Faure sequence, $\phi_n = (\phi_n^1, \phi_n^2, ..., \phi_n^s)$, as follows:

Let b be the smallest prime number that is bigger than or equal to s. We start as before by rewriting n in base b as

$$n = (a_m^1 ... a_1^1 a_0^1)_b = \sum_{j=0}^{m} a_j^1 b^j. \tag{1.18}$$

The first component of vector ϕ_n is defined as:

$$\phi_n^1 = \sum_{j=0}^{m} \frac{a_j^1}{b^{j+1}}.$$

Then, the coefficients a_j^k of b^j in (1.18) are updated as follows

$$a_j^k = \sum_{i \geq j}^{m} C_j^i a_i^{k-1} \pmod b,\ s \geq k \geq 2,\ m \geq j \geq 0,$$

where $C_j^i = \dfrac{i!}{j!(i-j)!}$.

The remaining components of vector ϕ_n are obtained recursively by:

$$\phi_n^k = \sum_{j=0}^{m} \frac{a_j^k}{b^{j+1}},\ s \geq k \geq 2.$$

Table 1.1 displays the first ten elements of the three dimensional Faure sequence.

Table 1.1: First ten elements of the three dimensional Faure sequence.

n	a_0	a_1	a_2	ϕ_n^1	ϕ_n^2	ϕ_n^3
1	1	0	0	1/3	1/3	1/3
2	2	0	0	2/3	2/3	2/3
3	0	1	0	1/9	4/9	7/9
4	1	1	0	4/9	7/9	1/9
5	2	1	0	7/9	1/9	4/9
6	0	2	0	2/9	8/9	5/9
7	1	2	0	5/9	2/9	8/9
8	2	2	0	8/9	5/9	2/9
9	0	0	1	1/27	16/27	13/27
10	1	0	1	10/27	25/27	22/27

By convention, the word "low-discrepancy" sequence is used for a sequence with D_N^* given by $O(\log^s N/N)$. From the Koksma-Hlawka inequality, this implies the QMC error bound for numerical integration is $O(\log^s N/N)$ when low-discrepancy sequences are used.

1.3 Randomized quasi-Monte Carlo

A drawback of the QMC method is that it is not practical to assess the accuracy of its estimates. To overcome this drawback, researchers introduced randomized quasi-Monte Carlo methods (RQMC) where a statistical error analysis is available. These methods allow independent simulations via QMC, and the resulting estimates can be analyzed statistically. Suppose $\mathbf{u}$ is a random vector with the uniform distribution on I^s. Let $\beta_{\mathbf{u}} = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$, $\mathbf{x}_i \in I^s$ for $i = 1, 2, ...$, be low discrepancy sequence indexed by $\mathbf{u}$. For each sequence, we have the estimate

$$Q(\beta_{\mathbf{u}}) = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i). \tag{1.19}$$

Then, $Q = \int_{I^s} f(\mathbf{x})d\mathbf{x}$ is estimated by taking the average of M samples

$$Q \simeq \frac{1}{M}\sum_{m=1}^{M} Q(\beta_{\mathbf{u}_m}). \tag{1.20}$$

RQMC has three general properties:

1. $E[Q(\beta_{\mathbf{u}})] = Q$
2. $Var(Q(\beta_{\mathbf{u}}) = O(N^{-2}\log^{2s}(N))$

3. $|Q(\beta_{\mathbf{u}}) - Q| \leq V(f)D^*(\beta_{\mathbf{u}})$

A detailed survey of RQMC methods is given by Ökten and Eastman [16]. We next discuss some examples of RQMC sequences.

1.3.1 Random-start Halton sequences

We first present an alternative description of the Halton sequence based on the von Neuman-Kakutani transformation [32],[60], [64]. We next define the rightward carry addition $\oplus$. Let $b \geqslant 2$ be an integer. For $x \in [0,1]$, the representation of x in base b is

$$x = \sum_{k=0}^{\infty} \frac{u_k}{b^{k+1}}.$$

If $x = 1$, set $x = .(b-1)(b-1)....$ (in base b). The rightward carry sum in base b of x and $1/b$ is

$$T_b(x) = x \oplus \frac{1}{b} = \frac{1+u_m}{b^{m+1}} + \sum_{k \geq m} \frac{u_k}{b^{k+1}},$$

where $m = \min\{k | u_k \neq b-1\}$.

The operator $T_b(x)$ is called the b-adic von Neumann-Kakutani transformation. The sequence $\{T_b^n(x)\}_{n=0}^{\infty}$ is defined recursively by

$$T_b^0(x) = x$$

$$T_b^{n+1} = T_b^n(T_b^{n-1}(x)), n \geq 1.$$

Note that if $x = 0$, $\{T_b^n(0)\}_{n=0}^{\infty}$ is the van der Corput sequence in base b. If $x = .d_0...d_j$ (in base b), denote the corresponding integer $m = d_j...d_0$ (in base b), then we have $x = \phi_b(m)$ and $T_b^n(x) = \phi_b(m+n)$, where $\phi_b(m)$ is the *radical-inverse* function defined in Section 1.2.4

The von Neumann-Kakutani transformation can be generalized to higher dimensions. For $\mathbf{x} = (x_1, ..., x_s) \in I^s$, let $\mathbf{b} = (b_1, ..., b_s)$ be a vector in $\mathbb{N}^s$, where $b_1, ..., b_s$ are pairwise prime, and define the s-dimensional sequence $T_{\mathbf{b}}^n(\mathbf{x}) = \{(T_{b_1}^n(x_1), ..., T_{b_s}^n(x_s))\}_{n=0}^{\infty}$. Using the von Neumann-Kakutani transformation on I^s, we have the following definition of random-start Halton sequence. Let $\mathbf{x}$ be a random vector with uniform distribution on I^s, then the sequence $\{T_{\mathbf{b}}^n(\mathbf{x})\}_{n=0}^{\infty}$ is called a random-start Halton sequence. The following theorem (see [33]) shows that the discrepancy of the random-start Halton sequence is $O((\log^s N)/N)$.

Theorem 8 *For any start point $\boldsymbol{x} \in I^s$, the sequence $\omega = \{T_{\boldsymbol{b}}^n(\boldsymbol{x})\}_{n=0}^{\infty}$ is a low-discrepancy sequence with the star-discrepancy*

$$D_N^*(\omega_N) \leq \frac{1}{N}\left[1 + \prod_{i=1}^{s}(b_i - 1)\frac{\log(b_i N)}{\log b_i}\right]. \tag{1.21}$$

Note that the upper bound of the star discrepancy does not depend on the start point $\mathbf{x}$.

1.3.2 Scrambled (t, m, s)-nets and (t, s)-sequences

The (t, m, s)-nets and (t, s)-sequences are special constructions of QMC point sets and sequences. A detailed survey of nets and sequences is given by Niederreiter [46]. Here we review some basic definitions. Let $b \geq 2$ be an integer. A subinterval E of I^s of the form

$$E = \prod_{i=1}^{s}[a_i b^{-d_i}, (a_i + 1)b^{-d_i}),$$

with $a_i, d_i \in \mathbb{Z}, d_i \geq 0, 0 \leq a_i < b^{d_i}$ for $1 \leq i \leq s$ is called an elementary interval in base b.

Definition 4 *Let $0 \leq t \leq m$ be integers. A (t, m, s)-net in base b is a point set P of b^m points in I^s such that $A(E; P) = b^t$ for every elementary interval E in base b with $\lambda_s(E) = b^{t-m}$.*

Definition 5 *Let $t \geq 0$ be an integer. A sequence $\boldsymbol{x}_0, \boldsymbol{x}_1, ...$ of points in I^s is a (t, s)-sequence in base b if , for all integers $k \geq 0$ and $m > t$, the point set consisting of the $\boldsymbol{x}_n$ with $kb^m \leq n < (k + 1)b^m$ is a (t, m, s)-net in base b.*

Owen [53] introduced a way of randomizing (t, m, s) nets and (t, s) sequences. The scrambled nets provide improvements for $Var(Q(\beta_{\mathbf{u}}))$ if the integrand is smooth enough. For example, for a smooth function f, Owen [54] shows that $Var(Q(\beta_{\mathbf{u}})) = O(N^{-3}\ln^{s-1}(N))$. However, Owen's scrambling requires a great many permutations.

1.3.3 Random shifting

Random shifting is a simple method and can be applied to any QMC sequence. Suppose we have a QMC sequence $\omega = \{\mathbf{x}_1, \mathbf{x}_2, ...\}, \mathbf{x}_i \in I^s$. In the random shifting method, we generate a random vector $\mathbf{u} \in I^s$, then we construct a RQMC sequence $\omega^{(\mathbf{u})} = \{\mathbf{x}_1^{(\mathbf{u})}, \mathbf{x}_2^{(\mathbf{u})}, ...\}$, where $\mathbf{x}_i^{(\mathbf{u})} = (\mathbf{x}_i + \mathbf{u})$ mod 1, $i = 1, 2, ...$

Theorem 9 *The discrepancy of a shifted low-discrepancy sequence satisfies*

$$D_N(\omega_N^{(\boldsymbol{u})}) \leq 2^s D_N(\omega_N).$$

More properties and applications of random shifting are considered in [16].

1.4 Monte Carlo methods in computational finance

The Monte Carlo method is used widely in pricing of derivative securities and risk management. The fundamental implication of asset pricing theory states that the price of a derivative security can be represented as an expectation under certain circumstances (see Glasserman [26]). Evaluating a derivative security by the MC method involves simulating paths of stochastic processes. These paths are generated by sampling random points from a space of paths. The dimension and distribution of the space depend on the model used for pricing the derivative. For a high dimensional problem, the MC method in general is more appropriate than a finite difference method.

Using the MC method, one should consider two important factors: the variance, or the risk, of the estimation (equation (1.5)) and the computing time. The efficiency of the MC method can be improved by using better random number generators to reduce the computing time and the variance. Quasi Monte Carlo method is also used to reduce the MC simulation error. We will discuss more details about these techniques in Chapter 2. Modifying model's assumptions is another way to improve the efficiency of a financial model. In this section, we introduce the variance gamma model, a modification of the famous Black-Scholes model, for option pricing.

Black and Scholes [11] developed an important mathematical model for option pricing in 1973. The model, called the Black-Scholes model, was the first *quantitative* tool to price options, and has been used in financial industry since. Option traders derive option prices from the Black-Scholes model and then compare it with the prevailing option price in the exchange in order to determine if a particular option contract is over or under valued, hence assisting them in their options trading decision. However, some unrealistic assumptions of the model lead to some disagreements of the theoretical values with those from the reality, e.g., the volatility smile. One of the specific assumptions that are known to relate to this problem is that the sample paths of the diffusion process are *continuous* functions of time. In fact, as noted by [5], a jump component in modeling option price dynamics is important to overcome the difficulties in explaining the volatility smile effects, for example, in short-dated option prices.

In 1990, Madan and Seneta [39] introduced the variance gamma (VG) process as a model for option pricing. The improvement in the VG model is that there is no continuous martingale component, allowing it to overcome some shortcomings of the Black-Scholes model. In particular, the VG process is a pure jump process that has infinite number of jumps in each interval of time. The VG model is a promising method to price options because it allows for a wider modeling of skewness and kurtosis than the Brownian motion does. In its original version, the VG model is introduced as a time-changed Brownian motion without drift by a gamma process [39]. In 1998, Madan, Carr, and Chang [38] extended this model for a Brownian motion with constant drift and volatility at a random time change given by a gamma process. In these works, the authors developed a closed-form solution for the European option price, which gives better approximations to the market option prices when compared to the original Black-Scholes model.

The VG process can also be defined as the difference of two increasing gamma processes. The gamma process with the positive sign describes the increase of the price, while the gamma process with the negative sign expresses the decrease of the price. In the VG model, each unit of time is viewed as an economically relevant time length given by an independent random variable which follows the gamma distribution of unit mean and positive variance. The VG model has two new parameters compared with the Black-Scholes model; one controls the kurtosis, and the other controls the skewness of the distribution.

Several modified versions of the VG model have been developed. Since the VG process has a closed-form expression for its characteristic function, Carr and Madan [12] were able to use the fast Fourier transform to calculate European option prices based on the VG model. This approach, however, does not fit well with the path-dependent options. Wang [62] has developed a version of the VG model that decomposes the marginal VG processes into independent components, and discussed its applications in pricing multi-asset options, e.g., exchange options, spread options, basket options and cross-currency options. Sensitivity analysis of options using the VG model is discussed in [10].

Monte Carlo and quasi-Monte Carlo methods are alternative approaches to compute option prices under the VG model in both path-independent and path-dependent cases, and improving the efficiency of these methods is of interest to many researchers. For example, Ribeiro and Webber [58] developed the QMC method with bridge sampling for path-dependent options under the VG

model. Avramidis and L'Ecuyer [2] introduced efficient MC and QMC algorithms for the option prices under the VG model.

1.5 Markov chains

In this section we introduce a special kind of stochastic process, a Markov chain, where the outcome of an experiment depends only on the outcome of the previous experiment. We later discuss hidden Markov models in Chapter 5.

A. A. Markov (1856-1922) introduced Markov chains in 1906 when he gave the first theoretical results for stochastic processes by using the term "chain" for the first time. In 1913 he used Markov chains to calculate letter sequences of the Russian language. A Markov chain includes three components: a sequence of random variables $X = (X_0, X_1, ...)$ (the chain), a state space $S = (S_1, S_2, ...)$ (in which the random variables take values), and a transition matrix.

Definition 6 *A discrete-time stochastic process $X = (X_t),\ t = 0, 1, ...,$ taking values in a finite state space $S = (S_i),\ i = 1, 2, ..., N$ is said to be a (homogenous) Markov chain if it satisfies the Markov property and stationary transition probabilities:*

1. *The conditional distribution of X_t given $X_0, ..., X_{t-1}$ is the same as the conditional distribution of X_t given X_{t-1} only*
2. *The conditional distribution of X_t given X_{t-1} does not depend on t.*

By definition, a Markov chain process starts in one of these states in the state space S and moves successively from one state to another. Each move is called a step. If the chain is currently in state S_i, then it moves to state S_j at the next step with a probability that does not depend upon which states the chain was before. More specifically, a Markov chain has a constant matrix of transition probabilities (or transition matrix), denoted by $A = (a_{ij}),\ i, j = 1, 2, ..., N$, where

$$a_{ij} = P(X_t = S_j | X_{t-1} = S_i).$$

The transition matrix satisfies:

$$a_{ij} \geq 0 \ i, j = 1, 2, ..., N$$

and

$$\sum_{j=1}^{N} a_{ij} = 1.$$

Given an initial vector of distributions at the initial time $p = (p_i),\ i = 1, 2, ..., N$, where $p_i = P(X_0 = S_i)$, the matrix A allows us to compute the distribution at any subsequent time. For example, $P(X_1 = j, X_0 = S_i) = p_i a_{ij}$.

Theorem 10 *Let A be the transition matrix of a Markov chain, and let p be the probability vector which represents the initial distribution. Then the probability that the chain is in state S_i after n steps is the i^{th} entry in the vector*

$$p\underbrace{AA...A}_{t\ times} = pA^t.$$

Theorem 11 *Let A be the transition matrix of a Markov chain. The ij^{th} entry of the matrix $A^n = \underbrace{AA...A}_{n\ times}$, denoted by $p_{ij}^{(n)}$, gives the probability that the Markov chain, starting in state S_i, will be in state S_j after n steps.*

More properties of Markov chains can be found in [49].

CHAPTER 2

THE ACCEPTANCE-REJECTION METHOD FOR LOW-DISCREPANCY SEQUENCES

2.1 Introduction

Model simulations involve generating pseudorandom numbers from various probability distributions used in the model. How do we generate a QMC sequence from a distribution $F(x)$? The process is somewhat similar to MC. One starts with a QMC sequence from the uniform distribution on $(0,1)^s$ and then applies a transformation method to the sequence in order to obtain a sequence from the target distribution. Currently, the only general transformation method used for QMC is the inverse transformation method (the Box-Muller method is also applicable in QMC [27], but its scope is smaller). The acceptance-rejection method is usually avoided in QMC, though "smoothed" versions of it were introduced by Moskowitz & Caflisch [4] and Wang [63]. The reasons for this avoidance has to do with some theoretical difficulties that involve the inapplicability of Koksma-Hlawka type inequalities to indicator functions with infinite variation.

If the inverse transformation method is computationally expensive for a particular distribution, then its application to a QMC sequence can make the overall QMC simulation too expensive to provide any advantages over the MC simulation. An example of costly inverse transformation algorithm appears in the simulation of the variance gamma model by QMC. Avramidis et. al. [3] comment on the additional cost of computing inverse of beta, gamma, and normal distributions, which are needed in the generation of the variance gamma model, and suggest that this additional cost needs to be considered while assessing the efficiency of different estimators.

In this chapter, we present a QMC version of the acceptance-rejection method, prove a convergence result, and develop error bounds. We present QMC algorithms based on acceptance-rejection for the normal, beta and gamma distributions. We also compare our acceptance-rejection QMC with the "smoothed" acceptance-rejection algorithms by Moskowitz & Caflisch [4], and Wang [63]. The availability of acceptance-rejection as a transformation method for QMC significantly broadens its scope.

2.2 The acceptance-rejection algorithm

The acceptance-rejection method is one of the standard methods used for generating distributions. Assume we want to generate from the density $f(x)$, and there is another density $g(x)$ (with CDF $G(x)$) we know how to sample from, say, by using the inverse transformation method. Assume the density functions $f(x), g(x)$ have the same domain, (a, b), and there exists a finite constant $C = \sup_{x\in(a,b)} f(x)/g(x)$. Let $h(x) = f(x)/Cg(x)$. The Monte Carlo acceptance-rejection algorithm is:

Algorithm 1 *Acceptance-rejection algorithm to generate pseudorandom numbers from the density* $f(x)$.

1. *Generate pseudorandom numbers* u, v *from the uniform distribution on* $(0, 1)$
2. *Generate* X *from* $g(x)$ *by* $X = G^{-1}(u)$
3. *If* $v \leq h(X)$ *accept* X*; Otherwise reject* X
4. *Repeat Steps 1 to 3, until the necessary number of points have been accepted.*

Acceptance-rejection is usually avoided in QMC because it involves integration of a characteristic function: this is the step that corresponds to accepting a candidate by a certain probability. Since characteristic functions can have infinite variation in the sense of Hardy and Krause, and since the celebrated Koksma-Hlawka inequality [46] links the integration error to the variation of the integrand, researchers for the most part have stayed away from the acceptance-rejection method with low-discrepancy sequences. Two notable exceptions are Moskowitz and Caflisch [4] and Wang [63]. In these papers, smoothed versions of acceptance-rejection are introduced. These methods replace the characteristic functions by continuous ones, thereby removing functions with infinite variation. However, these smoothing methods can be very time consuming; if one considers efficiency (time multiplied by error), the smoothing method can be worse than crude MC simulation. We will present such examples in Section 2.4. Perhaps for this reason, the smoothing methods have not gained much ground in applications.

For MC, acceptance-rejection is a very powerful tool. There are several specialized algorithms that combine acceptance-rejection with other techniques to obtain fast simulation methods for

many distributions used in computing; for a recent reference see Fishman [22]. Currently, the QMC method cannot be effectively used in these algorithms, since the smoothing techniques are expensive.

Let $\{x_1, ..., x_N\}$ be numbers obtained from a QMC algorithm that generates the distribution function $F(x)$. How well these numbers approximate $F(x)$ is given by the F-star discrepancy of $\{x_1, ..., x_N\}$:

$$D_F^*(x_1, ..., x_N) = \sup_{\alpha \in [a,b]} \left| \frac{A([a, \alpha); \{x_1, ..., x_N\})}{N} - F(\alpha) \right|$$

where (a, b) is the support of F, and the function $A([a, \alpha); \{x_1, ..., x_N\})$ counts how many numbers in $\{x_1, ..., x_N\}$ belong to the interval $[a, \alpha)$. If F is the uniform distribution, we simply write $D^*(x_1, ..., x_N)$ and call it star discrepancy. Note that F-star discrepancy is the Kolmogorov-Smirnov statistic that measures the distance between the empirical and theoretical distribution functions. In our numerical results we will use the Anderson-Darling statistic which is a generalization of the Kolmogorov-Smirnov statistic (see [15]). The Anderson-Darling statistic corresponds to the "weighted" F-star discrepancy of a point set. More on the weighted discrepancy and corresponding Koksma-Hlawka type error bounds can be found in Niederreiter & Tichy [48] and Ökten [50].

Next we introduce the acceptance-rejection method for low-discrepancy sequences.

Algorithm 2 *QMC Acceptance-rejection algorithm to generate a sequence whose F-star discrepancy converges to zero.*

1. *Generate a low-discrepancy sequence ω from the uniform distribution on $(0, 1)^2$*

$$\omega = \{(u_i, v_i) \in (0, 1)^2, i = 1, 2, ...\}$$

2. *For $i = 1, 2, ...$*

   - *Generate X from $g(x)$ by $X = G^{-1}(u_i)$*
   - *If $v_i \leq h(X)$ accept X; otherwise reject X*

3. *Stop when the necessary number of points have been accepted.*

The algorithm starts with a point set in $(0, 1)^2$

$$\omega_N = \{(u_i, v_i), i = 1, ..., N\}$$

and then applies inversion (Step 2) to obtain the new point set

$$P = \{(G^{-1}(u_i), v_i), i = 1, ..., N\}.$$

Assume $\kappa(N)$ points are accepted at Step 2 of the algorithm. After a renumbering of the indices, we obtain the set of "accepted points" in (a, b):

$$Q_{\kappa(N)} = \{G^{-1}(u_1), ..., G^{-1}(u_{\kappa(N)})\}. \tag{2.1}$$

The next theorem shows that the accepted points have F-star discrepancy that goes to zero with N. This result generalizes Theorem 2.4 of Wang [63] who proves a similar convergence result when the density $g(x)$ is the uniform density on $(0, 1)^s$, and $f(x)$ is a density function on $(0, 1)^s$.

Theorem 12 *We have*

$$D_F^*(Q_{\kappa(N)}) \to 0 \text{ as } N \to \infty \tag{2.2}$$

where $D_F^*(Q_{\kappa(N)})$ is the F-star discrepancy of the point set $Q_{\kappa(N)}$.

Proof

We need to prove that for any $\alpha \in (a, b)$

$$|F_{\kappa(N)}(\alpha) - F(\alpha)| = \left| \frac{A([a, \alpha); Q_{\kappa(N)})}{\kappa(N)} - F(\alpha) \right| \to 0, \tag{2.3}$$

where $F_{\kappa(N)}(\alpha)$ is the empirical CDF. Define the set

$$\begin{array}{rl} E(\alpha) & = \left\{(x, y) \in (0, 1)^2 : G^{-1}(x) < \alpha, y \le h(G^{-1}(x))\right\} \\ & = \left\{(x, y) \in (0, 1)^2 : x < G(\alpha), y \le h(G^{-1}(x))\right\} \end{array} \tag{2.4}$$

(for simplicity we will assume $G(x)$ is strictly increasing). Consider a point $G^{-1}(u_i) \in (a, b), i \in \{1, ..., N\}$. This point belongs to $Q_{\kappa(N)}$ and falls into $[a, \alpha)$ if and only if

1. $G^{-1}(u_i) < \alpha$,
2. $G^{-1}(u_i)$ is accepted in Step 2, i.e., $(G^{-1}(u_i), v_i) \in P$ is such that $v_i \le h(G^{-1}(u_i))$.

Therefore, $G^{-1}(u_i) \in [a, \alpha), i \in \{1, ..., N\}$, if and only if $(u_i, v_i) \in E(\alpha)$, which implies

$$A([a, \alpha); Q_{\kappa(N)}) = A(E(\alpha); \omega_N).$$

Now, we work on the local discrepancy:

$$\begin{aligned} &\left|\frac{A([0,\alpha);Q_{\kappa(N)})}{\kappa(N)} - F(\alpha)\right| \\ &= \left|\frac{N}{\kappa(N)}\frac{A(E(\alpha);\omega_N)}{N} - \frac{N}{\kappa(N)}Vol(E(\alpha)) + \frac{N}{\kappa(N)}Vol(E(\alpha)) - F(\alpha)\right| \\ &\leq \frac{N}{\kappa(N)}\left|\frac{A(E(\alpha);\omega_N)}{N} - Vol(E(\alpha))\right| + \left|\frac{N}{\kappa(N)}Vol(E(\alpha)) - F(\alpha)\right|. \end{aligned} \tag{2.5}$$

Here $Vol(E(\alpha))$ refers to the Lebesgue measure of the set $E(\alpha)$. Note that ω_N is a u.d. mod 1 sequence in $(0,1)^2$, and the boundary of the set $E(\alpha)$ has Lebesgue measure zero since $h(G^{-1}(x))$ is a continuous function on $(0,1)$. Thus, we have:

$$\left|\frac{A(E(\alpha);\omega_N)}{N} - Vol(E(\alpha))\right| \to 0 \tag{2.6}$$

as $N \to \infty$. Substituting $\alpha = b$ in (2.5), we obtain

$$\frac{A(E(b);\omega_N)}{N} \to Vol(E(b)).$$

Indeed, note that (u_i, v_i) from ω_N belongs to $E(b)$ if and only if $v_i \leq h(G^{-1}(u_i))$, which gives us all the accepted points, i.e., $A(E(b);\omega_N) = \kappa(N)$. Then, we have

$$\frac{\kappa(N)}{N} \to Vol(E(b)). \tag{2.7}$$

Equations (2.6) and (2.7) imply the first term of the upper bound of inequality (2.5) converges to zero. To prove that the second term also goes to zero, it suffices to show that

$$\frac{Vol(E(\alpha))}{Vol(E(b))} - F(\alpha) = 0. \tag{2.8}$$

From (2.4) we have

$$Vol(E(\alpha)) = \int_0^{G(\alpha)} \int_0^{h(G^{-1}(x)} dydx = \int_0^{G(\alpha)} h(G^{-1}(x))dx. \tag{2.9}$$

Change of variables yields: $u = G^{-1}(x)$, $du = dx/G'(G^{-1}(x))$, and thus

$$Vol(E(\alpha)) = \int_a^{\alpha} h(u)G'(u)du = \int_a^{\alpha} h(u)g(u)du = \frac{1}{C}\int_a^{\alpha} f(u)du = \frac{F(\alpha)}{C}. \tag{2.10}$$

Similarly, we have

$$Vol(E(b)) = \frac{1}{C}\int_a^{b} f(u)du = \frac{1}{C}, \tag{2.11}$$

since f is the density function on (a,b). This completes the proof.

Note that Theorem 12 generalizes to the case when X is an s-dimensional random vector in a straightforward way. In Algorithm 2, the low-discrepancy sequence ω would be replaced by an $(s+1)$-dimensional sequence

$$\omega = \{(\mathbf{u_i}, v_i) \in (0,1)^{s+1}, i = 1, 2, ...\}$$

where $\mathbf{u_i} \in (0,1)^s$.

2.3 Error bounds

The Koksma-Hlawka inequality does not provide a practical method to estimate error because both $V(f)$ and $D_N^*(\omega)$ are difficult to compute accurately. Several researchers introduced QMC error bounds for integrals without using the variation of the integrand. Niederreiter [47] established a QMC error bound for $(\mathcal{M}, \mu)$-uniform point sets. Hickernell and Wang [64] introduced the average integration error using random start Halton sequences. In this section, we discuss the error bounds of Niederreiter [47] and Hickernell & Wang [64].

2.3.1 MC-QMC integration of BVHK and UBVHK functions

Although the Koksma-Hlawka inequality (1.15) is not valid for functions of infinite variation, in practice, QMC estimation can be successfully used for such functions.

We consider Owen's classification of BVHK and UBVHK functions [55]. For integers $s \geq 1$ and $r \geq 0$, let $f_{s,r}$ be a function on I^s defined by

$$f_{s,r}(x) = \begin{cases} \max(x_1 + ... + x_s - 1/2, 0)^s, & r > 0 \\ c_{x_1+...+x_s>1/2}, & r = 0. \end{cases} \tag{2.12}$$

The following result is from Owen [55].

Theorem 13 *$V(f_{s,r})$ is finite for $s \leq r$ and infinite for $s \geq r + 2$.*

Therefore, with $s = 2$ and $r = 0$, the indicator function, c_E, is UBVHK, where $E = \{(x, y) \in I^2 | x + y > 1/2\}$. In contrast, the function c_E is BVHK for $E = \{(x, y) \in I^2 | y > 1/2\}$.

We will use MC and QMC methods to estimate the integral of two indicator functions. For each case, we compare the convergence of the simulations by plotting the actual error against the number of simulations. In Figure 2.1, the errors seem to converge to zero in both methods. In Figure 2.1(a), the integrand is a BVHK function, and in Figure 2.1(b), it is a UBVHK function. In

each case the QMC error is smaller than the MC error, and the variation of the function does not seem to have any effect on the convergence. In fact, to the best of our knowledge, in the literature there is no example of an integrand that exhibits a divergent error behavior due to its infinite variation.

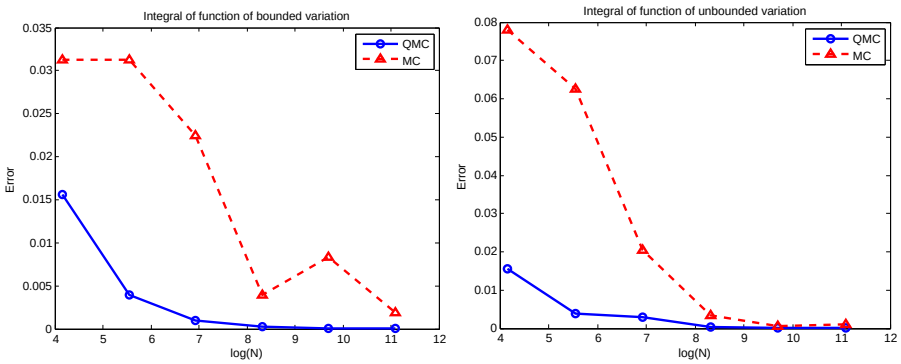

Figure 2.1: (a) $E = \{(x, y) \in I^2 | y > 1/2\}$ (b) $E = \{(x, y) \in I^2 | x + y > 1/2\}$

2.3.2 Error bounds for QMC with uniform point sets

Indicator functions, unless some conditions are satisfied ([55]), have infinite variation and thus Koksma-Hlawka inequality cannot be used to bound their error. This has been the main theoretical obstacle for the use of low-discrepancy sequences in acceptance-rejection algorithms. As a remedy, smoothing methods ([4], [63]) were introduced to replace the indicator functions by smooth functions so that Koksma-Hlawka is applicable. In this section we present error bounds that do not require the bounded variation assumption, and allow the analysis of our QMC Acceptance-Rejection algorithm. In the following section, we will compare our algorithm with the smoothing approach numerically.

Consider a general probability space $(\mathcal{X}, \mathcal{B}, \mu)$, where $\mathcal{X}$ is an arbitrary nonempty set, $\mathcal{B}$ is a σ-algebra of subsets of $\mathcal{X}$, and μ is a probability measure defined on $\mathcal{B}$. Let $\mathcal{M}$ be a nonempty subset of $\mathcal{B}$. For a point set $\mathcal{P} = \{x_1, \ldots, x_N\}$ and $M \subseteq \mathcal{X}$, define $A(M; \mathcal{P})$ as the number of elements in $\mathcal{P}$ that belong to M. A point set $\mathcal{P}$ of N elements of $\mathcal{X}$ is called $(\mathcal{M}, \mu)$-uniform if

$$A(M; \mathcal{P})/N = \mu(M) \tag{2.13}$$

for all $M \epsilon \mathcal{M}$. The definition of $(\mathcal{M}, \mu)$-uniform point sets is due to Niederreiter [47] who developed error bounds when uniform point sets are used in QMC integration. A useful feature of these

bounds is that they do not require the integrand to have finite variation. The following result is from Göncü and Ökten [52]:

Theorem 14 *Let $\mathcal{M} = \{M_1, ..., M_K\}$ be a partition of $\mathcal{X}$ and f be a bounded μ-integrable function on a probability space $(\mathcal{X}, \mathcal{B}, \mu)$, then for a point set $\mathcal{P} = \{x_1, ..., x_N\}$ we have*

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_{\mathcal{X}} f d\mu \right| \leq \sum_{j=1}^{K} \mu(M_j)(G_j(f) - g_j(f)) + \sum_{j=1}^{K} \epsilon_{j,N} \max(|g_j(f)|, |G_j(f)|) \tag{2.14}$$

where $\epsilon_{j,N} = |A(M_j; \mathcal{P})/N - \mu(M_j)|$, $G_j(f) = \sup_{t \in M_j} f(t)$ and $g_j(f) = \inf_{t \in M_j} f(t)$, $1 \leq j \leq K$.

Theorem 14 provides a general error bound for any point set $\mathcal{P}$. If the point set is an $(\mathcal{M}, \mu)$-uniform point set then the second summation on the right hand side becomes zero and the result simplifies to Theorem 2 of Niederreiter [47]. Setting $f = 1_S$, the indicator function of the set S, in Theorem 14, we obtain a simple error bound for indicator functions:

Corollary 3 *Under the assumptions of Theorem 14, we have*

$$\left| \frac{A(S; \mathcal{P})}{N} - \mu(S) \right| \leq \sum_{j=1}^{K} \mu(M_j)(G_j(1_S) - g_j(1_S)) + \epsilon_{j,N}.$$

Now consider Algorithm 2 (QMC Acceptance-Rejection algorithm) where a low-discrepancy sequence is used to generate the point set $Q_{a(N)}$ (see (2.1)). We proved that $|A([a, \alpha); Q_{a(N)})/a(N) - F(\alpha)| \to 0$ as $N \to \infty$ in Theorem 12. Corollary 3 yields an upper bound for the error of convergence. Indeed, let $S = [a, \alpha)$ for an arbitrary $\alpha \in (a, b)$, $\mathcal{X}$ be the domain for the distribution function F, and μ the corresponding measure. We obtain the following bound:

$$\left| \frac{A([a, \alpha);\ Q_{a(N)})}{a(N)} - F(\alpha) \right| \leq \sum_{j=1}^{K} \mu(M_j)(G_j(1_{[a,\alpha)}) - g_j(1_{[a,\alpha)})) + \epsilon_{j,a(N)}. \tag{2.15}$$

If the point set $Q_{a(N)}$ happens to be an $(\mathcal{M}, \mu)$-uniform point set with respect to the partition, then the term $\epsilon_{j,a(N)}$ vanishes.

2.3.3 Error bounds for RQMC methods

Next, we discuss randomized quasi-Monte Carlo (RQMC) methods and another error bound that addresses the bounded variation hypothesis.

Let $\mathcal{F}$ be the class of real continuous functions defined on $[0,1)^s$ and equipped with Wiener sheet measure μ. Theorem 15 shows that the mean variance of $Q(\beta_{\mathbf{u}})$ under this measure is $O(N^{-2}(\log N)^{2s})$. Since a function $f(x)$ chosen from the Brownian sheet measure has unbounded variation with probability one, this result provides an alternative error analysis approach to classical Koksma-Hlawka inequality which requires the integrand to be of finite variation. This result was obtained by Wang and Hickernell [64] (Theorem 5, page 894) for random-start Halton sequences. However, their proof is valid for any RQMC method, as we show next.

Theorem 15 *The average variance of the estimator, $Q(\beta_{\boldsymbol{u}})$, taken over function set $\mathcal{F}$, equipped with the Brownian sheet measure $d\mu$, is:*

$$\int_{\mathcal{F}} E[Q(\beta_{\boldsymbol{u}}) - Q]^2 d\mu(f) = O(N^{-2}(\log N)^{2s}).$$

Proof

$$\begin{aligned}
\int_{\mathcal{F}} E[Q(\beta_{\mathbf{u}}) - Q]^2 d\mu(f) &= \int_{\mathcal{F}} \int_{I^s} (Q(\beta_{\mathbf{u}}) - Q)^2 du d\mu(f) \\
&= \int_{I^s} \int_{\mathcal{F}} (Q(\beta_{\mathbf{u}}) - Q)^2 d\mu(f) f du \\
&= \int_{I^s} (T_N^*(\mathbf{1} - \beta_{\mathbf{u}}))^2 du. \\
&\leq \int_{I^s} (D_N^*(\mathbf{1} - \beta_{\mathbf{u}}))^2 du,
\end{aligned} \tag{2.16}$$

where $\mathbf{1} - \beta_{\mathbf{u}} = \{\mathbf{1} - \mathbf{x}_1, \mathbf{1} - \mathbf{x}_2, ..., \mathbf{1} - \mathbf{x}_N\}$ and $(T_N^*(f))^2 = \int_{\mathcal{F}} E_{QMC}(f)^2 df$, (a result in Wozniakowski [65]). The last inequality is obtained by applying the inequality $T_N^*(\omega_n) \leq D_N^*(\omega_n)$, where $T_N^*(\omega_n)$ is defined by Definition 2 in [16]. We also have:

$$\begin{aligned}
\left| \frac{A(\omega_{\mathbf{u}}, [0, \mathbf{y}])}{N} - \prod_{i=1}^{s} y_i \right| &= \left| \frac{A(\beta_u, [\mathbf{1} - \mathbf{y}, \mathbf{1}])}{N} - \prod_{i=1}^{s} y_i \right| \\
&\leq D_N(\beta_u) \\
&\leq 2^s D_N^*(\beta_u).
\end{aligned} \tag{2.17}$$

The last inequality is obtained by applying Theorem 5 in Chapter 1. By taking the supremum of the left hand side of the above inequality over all closed interval of the form $[0, \mathbf{y}]$, we obtain

$$D_N^*(\omega_u) \leq 2^s D_N^*(\beta_u) = O(N^{-1}(\log N)^s).$$

Therefore,

$$\int_{\mathcal{F}} E[Q(\beta_{\mathbf{u}}) - Q]^2 d\mu(f) = O(N^{-2}(\log N)^{2s}). \tag{2.18}$$

This completes the proof.

In our numerical results that follow, we use random-start Halton sequences. Theorem 14 can be used to analyze error for both inverse transformation and acceptance-rejection implementations that we will discuss. Theorem 15 applies only for the inverse transformation implementations, since it is not known whether the accepted points given by the acceptance-rejection algorithm satisfy the discrepancy bound $O(N^{-1}(\log N)^s)$.

2.4 Smoothing

In this section we will compare the QMC Acceptance-Rejection Algorithm 2 with the smoothed acceptance-rejection algorithms by Moskowitz & Caflisch [4], and Wang [63]. The algorithms will be compared numerically in terms of efficiency, which is defined as sample variance times computation time. We will use the same numerical examples that were considered in [4] and [63].

Consider the problem of estimating the integral $Q = \int_{I^s} f(x)dx$ using the importance function $p(x)$

$$Q = \int_{(0,1)^s} \frac{f(x)}{p(x)} p(x)dx.$$

The MC estimator for Q is

$$\widetilde{Q} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}, x_i \sim p(x).$$

The standard acceptance-rejection algorithm, Algorithm 1, takes the following form for this problem:

Algorithm 3 *(AR)Acceptance-Rejection*

1. *Select* $\gamma \geq \sup_{x\in(0,1)^s} p(x)$
2. *Repeat until* N *points have been accepted:*
   - *Sample* $x_i \in (0,1)^s$, $y_i \in (0,1)$
   - *If* $y_i < \frac{p(x_i)}{\gamma}$, *accept* x_i
     Otherwise, reject x_i

The smoothed acceptance-rejection method of Moskowitz and Caflisch [4] introduces a weight function $w(x, y)$ such that

$$\int_0^1 w(x, y)dy = \frac{p(x)}{\gamma}, x \in (0, 1)^s, y \in (0, 1).$$

The weight function $w(x, y)$ is generated by the following algorithm we call SAR1.

Algorithm 4 *(SAR1) Smoothed acceptance-rejection by Moskowitz and Caflisch [4]*

1. *Select* $\gamma \geq \sup_{x\in(0,1)^s} p(x)$, *and* $0 < \sigma << 1$
2. *Repeat until weight of accepted points is within one unit of* N*:*
   - *Sample* $x_i \in (0, 1)^s$, $y_i \in (0, 1)$
   - *If* $y_i < \frac{p(x_i)}{\gamma} - \frac{1}{2}\sigma$ *set* $w = 1$
     Else if $y_i > \frac{p(x_i)}{\gamma} + \frac{1}{2}\sigma$ *set* $w = 0$
     Else set $w = \frac{1}{\sigma}(\frac{p(x_i)}{\gamma} + \frac{1}{2}\sigma - y_i)$

Wang [63] extended the SAR1 algorithm by choosing functions $A(x)$, $B(x)$ such that

$$0 \leq A(x) < p(x) < B(x) \leq \gamma, \ x \in (0, 1)^s, \gamma \geq \sup_{x\in(0,1)^s} p(x),$$

and setting the weight function using the following algorithm (which we call SAR2).

Algorithm 5 *(SAR2) Smoothed acceptance-rejection by Wang [63]*

1. *Select* $\gamma \geq \sup_{x\in(0,1)^s} p(x)$, *and functions* $A(x)$, $B(x)$ *such that*

$$0 \leq A(x) < p(x) < B(x) \leq \gamma, x \in (0, 1)^s$$

2. *Repeat until weight of accepted points is within one unit of* N*:*
   - *Sample* $x_i \in (0, 1)^s$, $y_i \in (0, 1)$
   - *If* $y_i < \frac{A(x_i)}{\gamma}$ *set* $w = 1$
     Else if $y_i \geq \frac{B(x)}{\gamma}$ *set* $w = 0$
     Else if $\frac{p(x_i)}{\gamma} \geq y_i > \frac{A(x_i)}{\gamma}$ *set* $w = 1 + \frac{(p(x_i)-B(x_i))(\gamma y_i - A(x_i))}{(B(x_i)-A(x_i))(p(x_i)-A(x_i))}$
     Else set $w = \frac{(p(x_i)-A(x_i))(\gamma y_i - B(x_i))}{(B(x_i)-A(x_i))(p(x_i)-B(x_i))}$

Now we consider the example used in [4] (Example 3, page 43) and [63]. The problem is to estimate the integral $Q = \int_{I^s} f(x)dx$, where $s = 7$ and

$$f(x) = \exp(1 - (\sin^2(\frac{\pi}{2}x_1) + \sin^2(\frac{\pi}{2}x_2) + \sin^2(\frac{\pi}{2}x_3))) \arcsin(\sin(1) + \frac{x_1 + ... + x_7}{200}).$$

The importance function is

$$p(x) = \frac{1}{C} \exp(1 - (\sin^2(\frac{\pi}{2}x_1) + \sin^2(\frac{\pi}{2}x_2) + \sin^2(\frac{\pi}{2}x_3))),$$

where

$$C = \int_{(0,1)^7} \exp(1 - (\sin^2(\frac{\pi}{2}x_1) + \sin^2(\frac{\pi}{2}x_2) + \sin^2(\frac{\pi}{2}x_3)))dx = (\int_0^1 \exp(-\sin^2(\frac{\pi}{2}x))dx)^3.$$

Three estimators are used:

- Crude Monte Carlo (CR):

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i),\ x_i \sim U((0,1)^s)$$

- Acceptance-Rejection (AR):

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i)/p(x_i),\ x_i \sim p(x),\ x_i \textit{ are accepted points}$$

- Smoothed Acceptance-Rejection (SAR1 and SAR2):

$$\frac{1}{N}\sum_{i=1}^{N^*} w(x_i, y_i) f(x_i)/p(x_i),$$

  where N^* is a positive integer such that $\sum_{i=1}^{N^*} w(x_i, y_i)$ is approximately N.

Table 2.1 displays the efficiency of the algorithms. We normalize the efficiency of the algorithms by the efficiency of the crude Monte Carlo algorithm. For example, the efficiency of the Acceptance-Rejection (AR) algorithm, Eff_{AR} is computed by

$$\mathit{Eff}_{AR} = \frac{\sigma_{CR}^2 \times t_{CR}}{\sigma_{AR}^2 \times t_{AR}}, \tag{2.19}$$

where σ_{CR} is the sample standard deviation of M estimates obtained using the crude Monte Carlo algorithm, and t_{CR} is the corresponding computing time. Similarly, the parameters σ_{AR} and t_{AR}

refer to the sample standard deviation and computing time for the Acceptance-Rejection (AR) algorithm.

Although we are primarily interested in how these algorithms compare when they are used with low-discrepancy sequences, for reference, we also report efficiencies when the algorithms are used with pseudorandom numbers. The first part of the table reports the Monte Carlo values (MC) where the pseudorandom sequence Mersenne twister [43] is used, and the second part reports the (randomized) quasi-Monte Carlo (RQMC) values where random-start Halton sequences ([51], [64]) are used.

In the numerical results, $M = 64$, $\sigma = 0.2$ in the algorithm SAR1, and $A(x) = 1/Ce^2$, $B(x) = e/C$ in the algorithm SAR2. We consider the same sample sizes N as in [4] so that our results can be compared with theirs. Table 2.1 reports the sample standard deviation and efficiency (in parenthesis) for each algorithm. Note that in our notation, larger efficiency values suggest the method is better.

Based on the numerical results in Table 2.1, we make the following conclusions. In QMC, the AR algorithm has better efficiency than the smoothed algorithms SAR1, 4,and SAR2, 5, by approximately factors between 2 and 28. A part of the improved efficiency is due to the faster computing time of the AR algorithm. However, the AR algorithm also provides lower standard deviation for all samples. In the case of MC, the AR algorithm has still better efficiency, but with a smaller factor of improvement.

Table 2.1: Comparison of acceptance-rejection algorithm AR with its smoothed versions SAR1 and SAR2, in terms of sample standard deviation and efficiency (in parenthesis).

N	MC				QMC			
	CR	SAR1	SAR2	AR	CR	SAR1	SAR2	AR
256	$3.1e^{-2}$	$8.8e^{-4}$	$8.8e^{-4}$	$3.2e^{-4}$	$2.7e^{-3}$	$7.0e^{-4}$	$7.3e^{-4}$	$1.5e^{-4}$
	(1)	(211)	(317)	(3972)	(57)	(266)	(255)	(7233)
1024	$1.6e^{-2}$	$2.7e^{-4}$	$2.3e^{-4}$	$1.4e^{-4}$	$6.2e^{-4}$	$2.1e^{-4}$	$2.2e^{-4}$	$7.8e^{-5}$
	(1)	(652)	(1252)	(5829)	(284)	(686)	(656)	(6359)
4096	$8.8e^{-3}$	$7.8e^{-5}$	$8.8e^{-5}$	$8.0e^{-5}$	$1.6e^{-4}$	$5.0e^{-5}$	$5.0e^{-5}$	$2.6e^{-5}$
	(1)	(2595)	(2600)	(5610)	(1430)	(3872)	(3872)	(20308)
16384	$3.9e^{-3}$	$3.6e^{-5}$	$3.5e^{-5}$	$4.2e^{-5}$	$4.2e^{-5}$	$1.3e^{-5}$	$1.2e^{-5}$	$9.6e^{-6}$
	(1)	(2492)	(2786)	(3900)	(3900)	(9900)	(12350)	(25594)

CHAPTER 3

GENERATING DISTRIBUTIONS USING THE ACCEPTANCE-REJECTION METHOD

3.1 Introduction

In this chapter we will present QMC algorithms based on acceptance-rejection for generating normal, beta, and gamma distributions, and numerically compare them with their counterparts based on the inverse transformation method. In all the numerical results, Mersenne twister [43] is used for Monte Carlo, and random-start Halton sequences ([51], [64]) are used for quasi-Monte Carlo.

3.2 Generating normal distribution

Generation of the normal distribution is arguably one of the most common procedures in simulation. There are many algorithms for generating the normal distribution, including various approximations for the inverse transformation method, the Box-Muller method, and a fast algorithm known as the ziggurat method by Marsaglia [42]. The ziggurat method involves the acceptance-rejection and composition methods. In this section, we will provide a QMC version of this method. We will also consider a "textbook example" for generating the normal distribution from the exponential distribution with parameter 1, exp(1), by acceptance-rejection (Ross [59], page 70). We provide a QMC version of this method in Algorithm 5.

To generate random variable Z from normal distribution, $Z \sim \mathcal{N}(0,1)$, first we generate $|Z|$, then let $Z = |Z|$ or $Z = -|Z|$ with probability $p = .5$. The density function of $|Z|$ is

$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-x^2/2},\ x > 0.$$

Using the acceptance-rejection method, we choose $g(x)$ as the density function of the exponential distribution,

$$g(x) = e^{-x},\ x > 0.$$

Thus, the constant $C = \sup_{x>0} \frac{f(x)}{g(x)} = \sqrt{2e/\pi}$, and the function $h(x) = e^{-(x-1)^2/2}$. The CDF of $g(x)$ is

$$G(x) = 1 - e^{-x},$$

and its inverse

$$G^{-1}(x) = -\log(1-x).$$

Therefore

$$h(G^{-1}(x)) = e^{-(\log(1-x)+1)^2/2}.$$

Algorithm 6 *QMC Acceptance-Rejection for generating* $X \sim \mathcal{N}(0,1)$

1. *Generate a low-discrepancy sequence* ω *on* $(0,1)^3$

$$\omega = \{(u_i, v_i, w_i) \in (0,1)^3, i = 1, 2, ...\}$$

2. *For* $i = 1, 2, ...$
   - *Generate* $X \sim \exp(1)$ *by setting* $X = -\ln(u_i)$
   - *If* $\ln v_i \leq -(X-1)^2/2$
     - *If* $w_i < .5$ *accept* X
     - *Otherwise accept* $-X$
   - *Otherwise reject* X
3. *Stop when the necessary number of points have been accepted.*

There is another method to generate from the normal distribution: the ziggurat method. This method was introduced by Marsaglia and Tsang [42].

Algorithm 7 *Ziggurat Algorithm*

1. *Generate a random integer* i *from* $\{0, 1, 2, ..., n-1\}$
2. *Generate a random number* U_1 *from* $U(0,1)$
   (a) *If* $i = 0$ *set* $X = \frac{U_1 V}{f(x_n)}$
      - *If* $X < x_n$ *return* X
      - *Else generate* X *from the tail, (Marsaglia [41])*
   (b) *Else set* $X = U_1 x_i$

- *If* $X < x_{i-1}$ *return* X
- *Else generate a random number* U_2 *from* $U(0,1)$*, if* $[f(x_{i-1} - f(x_i)]U_2 < f(X) - f(x_i)$ *return* X

3. *Go to step 1*

Some authors (see, for example, [35]) observed that the accuracy of the algorithm can be improved, at the expense of some computing time, by removing some time saving tricks used in [42] where a single random number was used in multiple decisions. We follow the approach of [35] in designing a QMC version of the ziggurat algorithm, which is given by Algorithm 8.

Algorithm 8 *QMC Ziggurat for generating* $X \sim \mathcal{N}(0,1)$

1. *Initialization:*

   - $N = 128$*, and* $V = 9.91256303526217e^{-3}$
   - $\{x_i\}$*:* $x_N = R = 3.442619855899$*, and* $x_i = f^{-1}(V/x_{i+1} + f(x_{i+1}))$ *for* $i = (N-1), (N-2), ..., 0$
   - $\{f_i\}$*:* $f_i = f(x_i)$*, for* $i = 0, ..., N$
   - $\{w_i\}$*:* $w_0 = V/F(R)$*, and* $w_i = x_i$ *for* $i = 1, 2, ..., N$
   - $\{k_i\}$*:* $k_0 = Rf(R)/V$*, and* $k_i = x_{i-1}/x_i$ *for* $i = 1, ..., N$

2. *Generate a low-discrepancy sequence* ω *on* $(0,1)^5$

$$\omega = \{(u_j^{(1)}, u_j^{(2)}, u_j^{(3)}, u_j^{(4)}, u_j^{(5)}), j = 1, 2, ...\}$$

3. *For* $j = 1, 2, ...$

   - *Set* $i =$ *Floor* $(u_j^{(1)} N)$
   - *Set* $U = 2u_j^{(2)} - 1$*,* $X = Uw_i$
   - *If* $|U| < k_i$ *return* X
   - *Else if* $i = 0$ *(generating* X *from the tail)*
     - *set* $X = -\log(u_j^{(3)})/R$*,* $Y = -\log(u_j^{(4)})$
     - *If* $2Y < X^2$
       - *If* $U > 0$ *return* $X + R$
       - *Else return* $-(X + R)$
   - *Else if* $(f_i + u_j^{(5)}(f_{i-1} - f_i)) < f(X)$ *return* X

4. Stop when the necessary number of points have been accepted.

Table 3.1 presents an empirical comparison of the original ziggurat method by Marsaglia [42] (Algorithm 7) with the Modified Ziggurat QMC algorithm (Algorithm 8) and the QMC Acceptance-Rejection algorithm (Algorithm 6). To compare the algorithms, we generate 10^6 random numbers from the normal distribution using each algorithm, and compute the Anderson-Darling statistic of the generated sample, and record the computing time. We define the efficiency of an algorithm by the product of the Anderson-Darling statistic and computing time. As before, we normalize efficiency by the original ziggurat algorithm. Table 3.1 reports the Anderson-Darling values, computing time, and relative efficiencies.

Table 3.1: $N = 1,000,000$ random numbers from standard normal distribution

Algorithms	Ziggurat	Modified Ziggurat (QMC)	AR (QMC)
A^2	1.226	0.095	0.004
Time (s)	0.02	0.21	0.32
Eff	1	1.23	17.13

We make the following observations:

- The Modified Ziggurat (QMC) method (Algorithm 8) has slightly better efficiency than the Ziggurat, by a factor of 1.23. This better efficiency is due to the fact that QMC offers better accuracy relative to computing time.
- Perhaps surprisingly, Acceptance-Rejection (QMC) method (Algorithm 6) has significantly better efficiency than Ziggurat, by about a factor of 17.

3.3 Generating beta distribution

The beta distribution, $\mathcal{B}(\alpha, \beta)$, has the density function

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathbf{B}(\alpha,\beta)},\ 0 < x < 1, \tag{3.1}$$

where $\alpha, \beta > 0$ are shape parameters, and $\mathbf{B}(\alpha, \beta)$ is the beta function,

$$\mathbf{B}(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt. \tag{3.2}$$

There are different algorithms for the generation of the beta distribution $\mathcal{B}(\alpha, \beta)$ depending on whether $\min(\alpha, \beta) > 1$, $\max(\alpha, \beta) < 1$, or neither. We will use the algorithm of Atkinson and Whittaker, *AW*, [22] for $\max(\alpha, \beta) < 1$, and the algorithm of Cheng [22] for $\min(\alpha, \beta) > 1$ for beta distribution. The algorithm uses a combination of composition, inverse transformation, and acceptance-rejection methods. We next introduce a QMC version of these algorithms.

3.3.1 Beta, $\mathcal{B}(\alpha, \beta)$, $\max(\alpha, \beta) < 1$

Atkinson and Whittaker [22] introduced the algorithm to generate variables from beta distribution, $\mathcal{B}(\alpha, \beta)$, with the assumption that the two beta parameters are less than 1.

Algorithm 9 *QMC-AW for generating* $X \sim \mathcal{B}(\alpha, \beta)$ *distribution where* $\max(\alpha, \beta) < 1$.

1. *Input parameters* α, β
2. *Set* $t = 1/[1 + \sqrt{\beta(1-\beta)/\alpha(1-\alpha)}]$, $p = \beta t/[\beta t + \alpha(1-t)]$
3. *Generate a low-discrepancy sequence* ω *on* $(0,1)^2$

$$\omega = \{(u_i, v_i) \in (0,1)^2, i = 1, 2, ...\}$$

4. *For* $i = 1, 2, ...$
   - *Set* $Y = -\log u_i$
   - *If* $v_i \leq p$
     - $X = t(v_i/p)^{1/\alpha}$
     - *If* $Y \geq (1-\beta)(t-X)/(1-t)$, *accept* X
     - *If* $Y \geq (1-\beta)\log((1-X)/(1-t))$, *accept* X
     - *Otherwise reject* X
   - *Otherwise*
     - $X = 1 - (1-t)[(1-u_i)/(1-p)]^{1/\beta}$
     - *If* $Y \geq (1-\alpha)(X/t - 1)$, *accept* X
     - *If* $Y \geq (1-\alpha)\log(X/t)$, *accept* X
     - *Otherwise reject* X
5. *Stop when the necessary number of points have been accepted.*

In Table 3.2 we consider several values for α and β that are less than one. **AR MC** and **AR QMC** in the table refer to the MC version of Algorithm AW, and its QMC version (Algorithm 9), respectively. The inverse transformation method[1] with MC and QMC is labeled as **Inverse MC** and **Inverse QMC**. We generate 10^5 numbers from each distribution, and record the computing time and the Anderson-Darling statistic of the sample, in Table 3.2.

Table 3.2: Comparison of inverse and acceptance-rejection algorithms, Algorithm AW (for MC) and Algorithm 9 (QMC-AW), in terms of the computing time and the Anderson-Darling statistic of the sample for the Beta distribution when $N = 10^5$ numbers are generated. The percentage points for the A^2 statistic at 5% and 10% levels are 2.49 and 1.93, respectively.

Algorithms		Inverse MC	AR MC	Inverse QMC	AR QMC
$\mathcal{B}(0.3, 0.3)$	Time(s)	0.32	0.04	0.32	0.04
	A^2	4.07e-1	1.35	1.13e-1	8.7e-4
$\mathcal{B}(0.3, 0.5)$	Time(s)	0.33	0.03	0.33	0.03
	A^2	1.67	7.65e-1	8.64e-2	2.24e-3
$\mathcal{B}(0.3, 0.7)$	Time(s)	0.34	0.03	0.34	0.03
	A^2	1.65	8.81e-1	1.33e-1	7.5e-4
$\mathcal{B}(0.5, 0.3)$	Time(s)	0.33	0.03	0.33	0.03
	A^2	2.77	8.39e-1	9.22e-2	6.4e-4
$\mathcal{B}(0.5, 0.5)$	Time(s)	0.32	0.03	0.32	0.02
	A^2	2.34	5.68e-1	2.1e-4	2.56e-3
$\mathcal{B}(0.5, 0.7)$	Time(s)	0.33	0.03	0.34	0.02
	A^2	6.24e-1	5.47e-1	2.5e-4	5.5e-4
$\mathcal{B}(0.7, 0.3)$	Time(s)	0.33	0.03	0.33	0.03
	A^2	6.86e-1	7.12e-1	1.35e-1	1.49e-3
$\mathcal{B}(0.7, 0.5)$	Time(s)	0.33	0.03	0.33	0.03
	A^2	1.92	2.15	2.7e-4	8.9e-4
$\mathcal{B}(0.7, 0.7)$	Time(s)	0.33	0.03	0.35	0.03
	A^2	8.99e-1	2.06	1.6e-4	5.7e-4

We make the following observations:

[1]The inverse transformation code we used is a C++ code written by John Burkardt (available at http://people.sc.fsu.edu/∼jburkardt/), and it is based on algorithms by Cran *et. al.* [14] and Majumder and Bhattacharjee [40]. The performance of the inverse transformation method greatly depends on the choice of tolerance for the method. A large tolerance can result in values that fail the Anderson-Darling goodness-of-fit test. A smaller tolerance increases the computing time. Therefore, in our numerical results, we set tolerances for different range of parameter values small enough so that the results pass the goodness-of-fit test. For α, $\beta < 1$, we set the tolerance to 10^{-8}.

1. The Acceptance-Rejection algorithm runs about 10 times faster than the inverse transformation algorithm, in both MC and QMC implementations. There is no significant difference in the computing times between MC and QMC, for each algorithm.

2. Inverse MC fails the Anderson-Darling test at the 5% level for $\mathcal{B}(0.5, 0.3)$. There are several Anderson-Darling values close to the 10% percentage point 1.93 for Inverse MC and AR MC methods. Switching to QMC improves the Anderson-Darling values significantly for both inverse and acceptance-rejection, implying better fit of the samples to the theoretical distribution. The Inverse QMC Anderson-Darling values range between 10^{-1} and 10^{-4}. The AR QMC Anderson-Darling values are more stable, and range between 10^{-3} and 10^{-4}.

3.3.2 Beta, $\mathcal{B}(\alpha, \beta)$, $\min(\alpha, \beta) > 1$

The BB algorithm of Cheng [22] used to generate random variables from the beta distribution generates beta variables from the second type, called $\mathcal{B}_2(\alpha, \beta)$, which has the density function

$$f_2(x) = \frac{x^{\alpha-1}}{\mathbf{B}(\alpha, \beta)(1+x)^{\alpha+\beta}}, \ x > 0 \tag{3.3}$$

Theorem 16 *If Y is a beta random variable of second type, $Y \sim \mathcal{B}_2(\alpha, \beta)$, then $X = \dfrac{Y}{1+Y}$ is a beta random variable, $X \sim \mathcal{B}(\alpha, \beta)$.*

Proof

We need to prove that $P(X \leq x^*) = F(x^*)$.

We have

$$P(X \leq x^*) = P(\frac{Y}{1+Y} \leq x^*) = P(Y \leq \frac{x^*}{1-x^*}) = F_2(\frac{x^*}{1-x^*}) = \int_{-\infty}^{\frac{x^*}{1-x^*}} f_2(y)dy.$$

By changing variable $x = \frac{y}{1+y}$, we have $y = \frac{x}{1-x}$, $dy = (1-x)^{-2}dx$, and

$$\int_{-\infty}^{\frac{x^*}{1-x^*}} f_2(y)dy = \int_{-\infty}^{x^*} f_2(\frac{x}{1-x})(1-x)^{-2}dx = \int_{-\infty}^{x^*} f(x)dx = F(x^*).$$

This completes the proof.

Using Theorem 16, Cheng [22] generates a random variable $Y \sim \mathcal{B}_2(\alpha, \beta)$, then returns $\dfrac{Y}{1+Y} \sim \mathcal{B}(\alpha, \beta)$. He uses the Burr XII density (in Section 3.2.1) as a proportional function to $f_2(x)$

$$g(x) = \lambda\mu\frac{x^{\lambda-1}}{(\mu+x^\lambda)^2}, \quad G(x) = \frac{x^\lambda}{\mu+x^\lambda}, \tag{3.4}$$

where $\mu = (\alpha/\beta)^{\lambda}$, and $\lambda = \sqrt{\dfrac{2\alpha\beta - (\alpha+\beta)}{\alpha+\beta) - 2}}$.
The acceptance-rejection constant is

$$C = \sup_{(0,\infty)} \frac{f_2(x)}{g(x)} = \frac{4\alpha^{\alpha}\beta^{\beta}}{\lambda \mathbf{B}(\alpha,\beta)(\alpha+\beta)^{\alpha+\beta}}. \tag{3.5}$$

The constant C is bounded by $4/e \sim 1.47$ for $\min(\alpha,\beta) > 1$. The acceptance-rejection function is

$$h(x) = \frac{x^{a-\lambda}(\mu + x^{\lambda})^2(\alpha+\beta)^{\alpha+\beta}}{4(x+1)^{\alpha+\beta}\alpha^{\alpha+\lambda}\beta^{\beta-\lambda}}. \tag{3.6}$$

Let $a = \alpha + \beta$, $b = \sqrt{(a-2)/(2\alpha\beta - a)}$, $\gamma = \alpha + b^{-1}$, and $u, v \sim (0,1)$.
Set $V = b\ln(u/(1-u))$, and $W = \alpha e^{V}$. We have $y = G^{-1}(u) \sim g(x)$, and the acceptance-rejection test $v \le h(y)$ is equivalent to $D \ge 0$, where

$$D = a\ln(a/(\beta+W)) + \gamma V - \ln 4 - \ln(u^2 v). \tag{3.7}$$

Since the calculations of $\ln(a/(\beta+W))$ and $\ln(u^2v)$ are slow, Cheng [22] uses the two preliminary tests. From the fact that $\ln(x)$ is a concave function

$$\theta x - \ln\theta - 1 \ge \ln x,\ \theta > 0.$$

Therefore,

$$\begin{aligned} D &= a\ln a - a\ln(b+W)) + \gamma V - \ln 4 - \ln(u^2v) \\ &\ge a\ln a - (\theta(\beta+W) - \ln\theta - 1) + \gamma V - \ln 4 - \ln(u^2v) \equiv D_1 \\ &\ge a\ln a - (\theta(\beta+W) - \ln\theta - 1) + \gamma V - \ln 4 - (\theta u^2 v - 1) \equiv D_2 \end{aligned}$$

Cheng [22] suggested the choice $\theta = 5$.

Algorithm 10 *BB^*, for generating $X \sim \mathcal{B}(\alpha,\beta)$ where* $\min(\alpha,\beta) > 1$

1. *Input beta parameters α, β*
2. *Set $a = \min(\alpha,\beta),\ b = \max(\alpha,\beta), c = a+b,\ d = \sqrt{(c-2)/(2ab-c)},\ \gamma = a + d^{-1}$*
3. *Generate a low-discrepancy sequence ω on I^2*

$$\omega = \{(u_i, v_i) \in I^2, i = 1, 2, ...\}$$

4. For $i = 1, 2, ...$

- *Set* $V = d\ln(u_i/(1-u_i))$, $W = ae^V$, $Z = u_i^2 v_i$, $R = \gamma V - 1.3862944$, $S = a + R - W$, *(the constant is* $\ln 4$*)*
  - *If* $S + 2.609438 \geq 5Z$, *accept (The constant is* $1 + \ln 5$*).*
  - *Set* $T = \ln Z$. *If* $S \geq T$, *accept*
  - *If* $R + a\ln(c/(b+W)) < T$, *reject*
- *If accept*
  - *If* $a = \alpha$ *return* $X = W/(b+W)$
  - *Otherwise return* $X = b/(b+W)$

In Table 3.3 we consider several values for α and β that are bigger than one. **AR MC** and **AR QMC** in the table refer to the MC version of Algorithm AW, and its QMC version (Algorithm 10), respectively. The inverse transformation method[2] with MC and QMC is labeled as **Inverse MC** and **Inverse QMC**. We generate 10^5 numbers from each distribution, and record the computing time and the Anderson-Darling statistic of the sample, in Table 3.3.

We make the following observations:

1. The Acceptance-Rejection QMC (Algorithm 10) is as fast as, or faster than, the MC implementation of Algorithm 7. The inverse transformation algorithms run about 10 times slower.

2. The Acceptance-Rejection QMC (Algorithm 10) has consistently the best efficiency. The efficiency improvements it provides over Inverse QMC can be as large as 6,099, although for one case when $\alpha = \beta = 0.5$, the efficiency of AR QMC and Inv QMC are very similar.

3.4 Generating the gamma distribution

The gamma distribution, $\mathcal{G}(\alpha, \beta)$, has the following property: if X is a random variable from $\mathcal{G}(\alpha, 1)$ then βX is the random variable from $\mathcal{G}(\alpha, \beta)$. Therefore, we only need algorithms to

[2]The inverse transformation code we used is a C++ code written by John Burkardt (http://people.sc.fsu.edu/∼jburkardt/), and it is based on algorithms by Cran et. al. [14] and Majumder and Bhattacharjee [40]. The performance of the inverse transformation method greatly depends on the choice of tolerance for the method. A large tolerance can result in values that fail the Anderson-Darling goodness-of-fit test. A smaller tolerance increases the computing time. Therefore, in our numerical results, we set tolerances for different range of parameter values small enough so that the results pass the goodness-of-fit test. For α, $\beta > 1$, we set the tolerance to 10^{-6}.

Table 3.3: Comparison of inverse and acceptance-rejection algorithms, Algorithm BB* (for $min(a,b) > 1$ and Algorithm 10 (QMC-AW), in terms of accuracy and efficiency for the Beta distribution when $N = 10^5$ numbers are generated.

Algorithms		Inverse MC	AR MC	Inverse QMC	AR QMC
$\mathcal{B}(1.5, 1.5)$	Time(s)	0.31	0.02	0.31	0.03
	A^2	.85449	1.12017	.00015	.00144
$\mathcal{B}(1.5, 2.0)$	Time(s)	.26	0.02	.27	0.02
	A^2	.56564	.80538	.00016	.00131
$\mathcal{B}(1.5, 2.5)$	Time(s)	.36	.02	.3	.03
	A^2	0.69054	.59326	.00025	.00463
$\mathcal{B}(2.0, 2.0)$	Time(s)	0.18	0.03	0.18	0.02
	A^2	2.12372	1.07203	.00016	.00075
$\mathcal{B}(2.0, 2.5)$	Time(s)	0.24	0.03	0.24	0.02
	A^2	.64079	.93179	.00012	.00096
$\mathcal{B}(2.0, 3.0)$	Time(s)	0.19	0.02	0.19	0.02
	A^2	1.69070	.36285	.00021	.00115
$\mathcal{B}(2.5, 2.0)$	Time(s)	0.24	0.02	0.24	0.02
	A^2	.85548	.30792	.00010	.00318
$\mathcal{B}(2.5, 2.5)$	Time(s)	0.29	0.02	0.29	0.02
	A^2	1.44800	.36973	.00012	.00101
$\mathcal{B}(2.5, 3.0)$	Time(s)	0.24	0.03	0.25	0.02
	A^2	.88369	.39129	.00012	.00146

generate random variables from $\mathcal{G}(\alpha, 1)$, which has the density function

$$f(x) = \frac{x^{\alpha-1}e^{-x}}{\mathbf{\Gamma}(\alpha)}, \qquad \alpha > 0,\ x \geq 0. \tag{3.8}$$

Here $\mathbf{\Gamma}$ is the gamma function

$$\mathbf{\Gamma}(z) = \int_0^\infty e^{-t}t^{z-1}dt, \tag{3.9}$$

where z is a complex number with positive real part.

We will consider two algorithms for generating the gamma distribution and present their QMC versions. The algorithms are:

- Algorithm CH by Cheng [13], which is applicable when $\alpha > 1$,
- Algorithm GS* by Ahrens-Dieter [22], which is applicable when $\alpha < 1$.

Next we introduce the QMC versions of these algorithms.

3.4.1 Gamma, $\mathcal{G}(\alpha, 1)$ with $\alpha > 1$

Cheng [13] developed the algorithm to generate variables from gamma distribution, $\mathcal{G}(\alpha, 1)$, for the case $\alpha > 1$.

Algorithm 11 *QMC-CH for generating $X \sim \mathcal{G}(\alpha, 1)$ distribution where $\alpha > 1$*

1. *Input gamma parameter α*
2. *Set $a = (2\alpha - 1)^{-1/2},\ b = \alpha - \log 4,\ c = \alpha + a^{-1}$*
3. *Generate a low-discrepancy sequence ω on $(0, 1)^2$*

$$\omega = \{(u_i, v_i) \in (0, 1)^2, i = 1, 2, ...\}$$

4. *For $i = 1, 2, ...$*
   - *Set $Y = a \log(\frac{u_i}{1-u_i}),\ X = \alpha e^Y,\ Z = u_i^2 v_i,\ R = b + cY - X$*
   - *If $R + 2.5040774 - 4.5Z \geq 0$, accept X*
   - *If $R \geq \log Z$, accept X*
   - *Otherwise reject X*
5. *Stop when the necessary number of points have been accepted.*

In Table 3.4 we consider several parameters for α in $\mathcal{G}(\alpha, 1)$ for α values greater than one. The parameters are chosen roughly in the range that was observed in the simulation of the variance gamma option pricing problem we will discuss later.

We generate 10^6 numbers from the gamma distribution using Algorithm 11 and compute the execution time of the algorithm and the Anderson-Darling statistic of the sample. The inverse transformation method[3] with MC and QMC is labeled as **Inverse MC** and **Inverse QMC**.

We make the following observations based on Table 3.4:

1. The AR QMC algorithm runs faster than the Inverse QMC algorithm by approximately factors between 30 and 48. Interestingly, the AR QMC algorithm is slightly faster than the AR MC algorithm for each case. Similarly, the AR MC is faster than Inverse MC, at about the same factors of speed up.

[3]The inverse transformation code we used is a C++ code written by John Burkardt, and it is based on two algorithms by Lau [34] and Mcleod [44]. Similar to the beta distribution, the performance of the inverse transformation method greatly depends on the choice for the tolerance parameter. In our numerical result we set the tolerance to 10^{-6} for $\alpha > 1$.

Table 3.4: Comparison of inverse and acceptance-rejection algorithms, Algorithm CH (for MC) and Algorithm 11 (QMC-CH), in terms of the computing time and the Anderson-Darling statistic of the sample for the Gamma distribution when $N = 10^6$ numbers are generated. The percentage points for the A^2 statistic at 5% and 10% levels are 2.49 and 1.93, respectively.

Algorithms		Inverse MC	AR MC	Inverse QMC	AR QMC
$\mathcal{G}(1.6,1)$	Time(s)	10.62	0.30	10.60	0.29
	A^2	2.90e-1	1.09	1.38e-4	8.6e-4
$\mathcal{G}(2.0,1)$	Time(s)	10.92	0.29	8.21	0.28
	A^2	8.78e-1	7.52e-1	1.93e-4	1.78e-3
$\mathcal{G}(2.4,1)$	Time(s)	12.32	0.29	11.74	0.28
	A^2	9.57e-1	1.83	1.0e-4	2.2e-4
$\mathcal{G}(2.8,1)$	Time(s)	12.41	0.28	13.17	0.27
	A^2	1.01	5.09e-1	1.1e-4	2.34e-3
$\mathcal{G}(3.2,1)$	Time(s)	12.62	0.28	12.64	0.27
	A^2	7.83e-1	5.67e-1	1.3e-4	1.21e-3

2. All samples pass the Anderson-Darling test at the 5% level. Switching to QMC drastically lowers the Anderson-Darling values: Inverse QMC values are around 10^{-4}, and AR QMC values range between 10^{-3} and 10^{-4}.

3.4.2 Gamma, $\mathcal{G}(\alpha,1)$ with $\alpha < 1$

We will present the algorithm written by Ahrens-Dieter [22] to generate variables from gamma distribution, $\mathcal{G}(\alpha,1)$, for $\alpha < 1$.

Algorithm 12 *QMC-GS* for generating $X \sim \mathcal{G}(\alpha,1)$ distribution where $\alpha < 1$*

1. *Input gamma parameter α*
2. *Set $b = (\alpha + e)/e$*
3. *Generate a low-discrepancy sequence ω on $(0,1)^3$*

$$\omega = \{(u_i, v_i, w_i) \in (0,1)^3, i = 1, 2, ...\}$$

4. *For $i = 1, 2, ...$*
   - *Set $Y = bu_i$*
   - *If $Y \leq 1$, set $X = Y^{1/\alpha}$*

- *Set* $W = -\log v_i$
- *If* $W \geq X$ *accept* X
- *Otherwise reject* X

- *Otherwise*
  - *Set* $X = -\log[(b - Y)/\alpha]$
  - *Set* $W = w_i^{1/(\alpha-1)}$
  - *If* $W \leq X$ *accept* X
  - *Otherwise reject* X

5. *Stop when the necessary number of points have been accepted.*

Similar to the Table 3.4, in Table 3.5, we consider several parameters for α less than one in $\mathcal{G}(\alpha, 1)$. We generate 10^6 numbers from the gamma distribution using Algorithm 12, and compute the execution time of the algorithm and the Anderson-Darling statistic of the sample. The inverse transformation method[4] with MC and QMC is labeled as **Inverse MC** and **Inverse QMC**.

Table 3.5: Comparison of inverse and acceptance-rejection algorithms, Algorithm GS* (for MC) and Algorithm 12 (QMC-GS*), in terms of the computing time and the Anderson-Darling statistic of the sample for the Gamma distribution when $N = 10^6$ numbers are generated. The percentage points for the A^2 statistic at 5% and 10% levels are 2.49 and 1.93, respectively.

Algorithms		Inverse MC	AR MC	Inverse QMC	AR QMC
$\mathcal{G}(0.2, 1)$	Time(s)	16.77	0.25	16.77	0.24
	A^2	1.73	5.18e-1	1.30	2.8e-4
$\mathcal{G}(0.4, 1)$	Time(s)	13.47	0.33	13.49	0.35
	A^2	6.76e-1	6.75e-1	4.28e-3	3.5e-4
$\mathcal{G}(0.6, 1)$	Time(s)	7.74	0.35	7.73	0.36
	A^2	1.64	1.01	5.15e-2	6.2e-4
$\mathcal{G}(0.8, 1)$	Time(s)	8.07	0.36	8.08	0.36
	A^2	5.18e-1	1.29	6.99e-3	3.1e-4

We make the following observations based on Table 3.5:

[4]The inverse transformation code we used is a C++ code written by John Burkardt, and it is based on two algorithms by Lau [34] and Mcleod [44]. Similar to the beta distribution, the performance of the inverse transformation method greatly depends on the choice for the tolerance parameter. In our numerical results, we set tolerances for different range of values for α small enough so that the results pass the goodness-of-fit test. For example, for $0.25 > \alpha \geq 0.20$, we set the tolerance to 10^{-14}. The convergence of the inverse transformation method was especially problematic for smaller α, for example, when $\alpha < 0.1$.

1. The AR QMC algorithm runs faster than the Inverse QMC algorithm by approximately factors between 22 and 70. For smaller α values, the convergence of the inverse transformation method is particularly slow.

2. All samples pass the Anderson-Darling test at the 5% level. As before, switching to QMC drastically lowers the Anderson-Darling values, especially for AR QMC whose values are around 10^{-4}. The Inverse QMC values range between 1.30 and 10^{-3}.

CHAPTER 4

THE VARIANCE GAMMA MODEL IN DERIVATIVE PRICING

Our main motivation is to develop fast and accurate QMC algorithms for the simulation of a particular Lévy process known as the variance gamma model ([38], [39]). This model is used in financial mathematics, and its QMC simulation is expensive due to the inverse transformation method. There are several ways to simulate the variance gamma model [24] and the methods involve generation of normal, beta, and gamma distributions.

4.1 Variance gamma process

In this section we discuss two ways to define the VG process; one as a Brownian motion with constant drift and volatility with a random time change, and one as the difference of two independent increasing gamma processes. The section is based on a paper by Madan, Carr and Chang [38].

4.1.1 Brownian motion

We first consider some concepts of the Brownian motion that is necessary for the introduction of the Gamma process in the subsequent sections.

Definition 7 *A stochastic process $B = \{B(t) = B(t;\theta,\sigma), t \geq 0\}$ is a Brownian motion with the drift parameter θ and the variance parameter σ if*

1. $B(0) = 0$.
2. *The process B has independent, stationary increments.*
3. $B(t+h)-B(t) \sim \mathcal{N}(\theta h, \sigma^2 h)$, *where* $\mathcal{N}(\mu,\sigma^2)$ *denotes the normal distribution with the density function* $f(x) = \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, $x \in \mathbb{R}$.

4.1.2 Gamma process

Definition 8 *A process $G = \{G(t) = G(t; \mu, \nu), t \geq 0\}$ is a Gamma process with mean rate $\mu > 0$ and variance rate $\nu > 0$ if*

1. *$G(0) = 0$.*
2. *The process G has independent, stationary increments.*
3. *$g_h = G(t+h) - G(t) \sim \mathcal{G}(\mu^2 h/\nu, \nu/\mu)$ for all $t \geq 0$, and $h > 0$, where $\mathcal{G}(\alpha, \beta)$ denotes the Gamma distribution with the density function given by*

$$f(z) = \frac{z^{\alpha-1} e^{-z/\beta}}{\beta^\alpha \mathbf{\Gamma}(\alpha)} \quad \text{for } \alpha, \beta > 0,\ z \geq 0.$$

The density of the increment $g_h = G(t+h) - G(t)$ is then given by

$$f_h(g) = \left(\frac{\mu}{\nu}\right)^{\frac{\mu^2 h}{\nu}} \frac{g^{\frac{\mu^2 h}{\nu}-1} \exp(-\frac{g\mu}{\nu})}{\mathbf{\Gamma}\left(\frac{\mu^2 h}{\nu}\right)}, g > 0 \tag{4.1}$$

where $\mathbf{\Gamma}(x)$ is the gamma function.

The gamma density has the characteristic function

$$\phi_{G(t)}(u) = E[\exp(iuG(t))] = \left(\frac{1}{1 - iu\frac{\nu}{\mu}}\right)^{\frac{\mu^2 t}{\nu}}. \tag{4.2}$$

For convenience, we just write g instead of g_h for the gamma time change, i.e, $g = g_h = G(t+h) - G(t)$.

4.1.3 Variance gamma as time-changed Brownian motion

The VG process $X(t; \sigma, \nu, \theta)$ is defined as

$$X(t; \sigma, \nu, \theta) = B(G(t; 1, \nu); \theta, \sigma)$$

where $B(t; \theta, \sigma)$ is a Brownian motion and $G(t; 1, \nu)$ is a Gamma process with a unit mean rate. The VG process, in other words, is a Brownian motion evaluated at a time given by a Gamma process. The density function of the VG process at time t is given by

$$f_{X(t)}(X) = \int_0^\infty \frac{1}{\sigma\sqrt{2\pi g}} \exp\left(-\frac{(X - \theta g)^2}{2\theta^2 g}\right) \frac{g^{\frac{t}{\nu}-1} \exp(-\frac{g}{\nu})}{\nu^{\frac{t}{\nu}} \mathbf{\Gamma}(\frac{t}{\nu})} dg. \tag{4.3}$$

The characteristic function for the VG process, $\phi_{X(t)}(u) = E[\exp(iuX(t))]$, is

$$\phi_{X(t)}(u) = \left(\frac{1}{1 - i\theta\nu u + \frac{\theta^2 \nu u^2}{2}}\right)^{\frac{t}{\nu}}. \tag{4.4}$$

4.1.4 Variance gamma as the difference of two gamma processes

The VG process can also be writen as the difference of two independent increasing gamma processes $G_p(t;\mu_p,\nu_p)$ and $G_n(t;\mu_n,\nu_n)$:

$$X(t;\sigma,\nu,\theta) = G_p(t;\mu_p,\nu_p) - G_n(t;\mu_n,\nu_n) \tag{4.5}$$

where

$$\begin{aligned}
\mu_p &= \tfrac{1}{2}\sqrt{\theta^2 + 2\sigma^2/\nu} + \tfrac{\theta}{2} \\
\mu_n &= \tfrac{1}{2}\sqrt{\theta^2 + 2\sigma^2/\nu} - \tfrac{\theta}{2} \\
\nu_p &= \mu_p^2\nu \\
\nu_n &= \mu_n^2\nu.
\end{aligned} \tag{4.6}$$

We can derive the formulas (4.6) by using the characteristic functions for two gamma processes (see (4.2)):

$$\phi_{G_p(t)}(u) = \left(\frac{1}{1 - iu\frac{\nu_p}{\mu_p}}\right)^{\frac{\mu_p^2 t}{\nu_p}},$$

and

$$\phi_{-G_n(t)}(u) = \left(\frac{1}{1 + iu\frac{\nu_n}{\mu_n}}\right)^{\frac{\mu_n^2 t}{\nu_n}}.$$

First, the characteristic function for the VG process is the product of the two characteristic functions

$$\phi_{X(t)}(u) = \left(\frac{1}{1 - iu\frac{\nu_p}{\mu_p}}\right)^{\frac{\mu_p^2 t}{\nu_p}} \left(\frac{1}{1 + iu\frac{\nu_n}{\mu_n}}\right)^{\frac{\mu_n^2 t}{\nu_n}}. \tag{4.7}$$

Then, from equations (4.4) and (4.7), the parameters must satisfy the equations

$$\begin{aligned}
\frac{\mu_p^2}{\nu_p} &= \frac{\mu_n^2}{\nu_n} = \frac{1}{\nu}, \\
\frac{\nu_p}{\mu_p} - \frac{\nu_n}{\mu_n} &= \theta\nu, \\
\frac{\nu_p}{\mu_p}\frac{\nu_n}{\mu_n} &= \theta^2\nu/2.
\end{aligned} \tag{4.8}$$

Finally, solving the system of equations in (4.8) gives the formulas in (4.6).

4.2 Simulating the variance gamma process

We consider two methods to simulate the VG process. The first method is to generate VG process as a time change Brownian motion (TCBM), and the second method is to generate VG process as the difference of two gamma processes (DTGP). Using the first method, based on the fact that the VG process conditioning on the increment of time change g_h, $g_h = G(t+h,1,\nu)-G(t,1,\nu)$, is Brownian motion with mean θg_h and variance $\sigma\sqrt{g_h}$, we first simulate the increment time change g_h from the $\mathcal{G}(h/\nu,\nu)$ distribution, then simulate $X(t)$ conditioned on g_h as the Brownian motion process. In each method, we will apply two techniques for simulating sample paths of the VG process: the sequential sampling and the bridge sampling. The algorithms given below are based on the algorithms in Avramidis and L'Ecuyer, [2], Deville, [19], Ribeiro and Webber, [58], and Fu [24].

4.2.1 Sequential sampling

Suppose that we need to generate the VG sample path $X(t)$ on [0,T] at discrete time points $0 = t_0 < t_1 < ... < T_{N-1} < T_N = T$. For simplicity, we consider uniform discretization, $t_i - t_{i-1} = \Delta_t = T/N$ for $i = 1$ to N. The sequential sampling method generates $X(t_1), X(t_2), ..., X(t_N)$ in sequence. In the TCBM method, we first generate the time change from the gamma process, $G(t,1,\nu)$,

$$g = G(t_i,1,\nu) - G(t_{i-1},1,\nu) \sim \mathcal{G}(\Delta_t/\nu,\nu).$$

Algorithm 13 for simulating the VG process by TCBM sequential sampling (TCBM(S)) is given next.

Algorithm 13 *TCBM(S)*

1. *Initialization:*
   - *VG parameters:* θ,σ,ν
   - *Size of sample path* N*, time to maturity* T
   - $X(t_0) = 0$, $\Delta_t = T/N$
2. *Simulation: for* $i = 1$ *to* N
   - *Generate* $g \sim \mathcal{G}(\Delta_t/\nu,\nu)$, $Z_i \sim \mathcal{N}(0,1)$
   - *Calculate* $X(t_i) = X(t_{i-1}) + \theta g + \sigma\sqrt{g}Z_i$

3. *Output:* $(X_t), t = t_0, ..., t_N$

The algorithm for simulating the VG process by DTGP sequential sampling (DTGP(S)) is described by Algorithm 14.

Algorithm 14 *DTGP(S)*

1. *Initialization:*
   - *VG parameters:* θ, σ, ν
   - *Size of sample path* N*, time to maturity* T
   - $X(t_0) = 0$, $\Delta_t = T/N$
   - *Calculate the parameters of two gamma processes* $\mu_p, \nu_p, \mu_n, \nu_n$ *using formula (4.6)*
2. *Simulation: for* $i = 1$ *to* N
   - *Generate* $g^+ \sim \mathcal{G}(\Delta_t \mu_p^2/\nu_p, \nu_p/\mu_p)$*;* $g^- \sim \mathcal{G}(\Delta_t \mu_n^2/\nu_n, \nu_n/\mu_n)$
   - *Return* $X(t_i) = X(t_{i-1}) + g^+ - g^-$
3. *Output:* (X_t), $t = t_0, ..., t_N$

4.2.2 Bridge sampling

For an arbitrary time t, $0 \le t_1 < t < t_2$, X_t can be sampled from the conditional distribution given ($X(t_1)$ and $X(t_2)$). For example, suppose we need to generate the VG sample path $X(t)$ on [0,T] at discrete time points $0 = t_0 < t_1 < ... < T_{N-1} < T_N = T$, where $N = 2^k$ for $k = 0, 1, 2, ...$ For simplicity, we assume the discretization is uniform; $t_i - t_{i-1} = \Delta_t = T/N$ for $i = 1$ to N. The bridge sampling can be done by generating the first two points at times t_0 and t_N. We then stratify successively at time $t_{N/2}$, $t_{3N/4}$, $t_{N/4}$, $t_{7N/8}$, $t_{5N/8}$, $t_{3N/8}$, $t_{N/8}$, and so on, until all points are generated.

We apply the property of conditional Brownian motion, $B(t) = B(t, \theta, \sigma)$, and gamma process, $G(t) = G(t, \mu, \nu)$. For an arbitrary time t, $0 \le t_1 < t < t_2$, the conditional distribution of $B(t)$ given $B(t_1)$ and $B(t_2)$ is the normal distribution $\mathcal{N}(aB(t_1) + (1-a)B(t_2), a(t-t_1)\sigma^2)$, where $a = \frac{t_2 - t}{t_2 - t_1}$. The conditional distribution of $G(t)$ given $G(t_1)$ and $G(t_2)$ is the same as $G(t_1) + [G(t_2) - G(t_1)]Y$, where $Y \sim \mathcal{B}(t-t_1)\mu^2/\nu, (t_2-t)\mu^2/v)$, $\mathcal{B}(\alpha, \beta)$ is the beta distribution.

The algorithm for simulating the VG process by TCBM bridge sampling (TCBM(B)) is presented in Algorithm 15.

Algorithm 15 *TCBM(B)*

1. *Initialization:*
   - *VG parameters:* θ, σ, ν*;*
   - *Size of sample path* $N = 2^k$*, time to maturity* T
   - $G(t_0) = 0$*;* $X(t_0) = 0$*,* $\Delta_t = T/N$
2. *Simulation:*
   - *Generate* $G(t_N) \sim \mathcal{G}(t_N/\nu, \nu)$
   - *Generate* $X(t_N) \sim \mathcal{N}(\theta G(t_N), \sigma^2 G(t_N))$
   - *For* $l = 1$ *to* k
     - $m = 2^{k-l}$
     - *For* $j = 1$ *to* 2^{l-1}
       - $i = (2j-1)m$
       - $d = (t_i - t_{i-m})/\nu$
       - *Generate* $Y \sim \mathcal{B}(d, d)$
       - $G(t_i) = G(t_{i-m}) + [G(t_{i+m} - G(t_i)]\sigma^2 Y$
       - *Generate* $Z \sim \mathcal{N}(0, [G(t_{i+m}) - G(t_i)]\sigma^2 Y)$
       - $X(t_i) = YX(t_{i+m}) + (1-Y)X(t_{i-m}) + Z$
3. *Output:* (X_t)*,* $t = t_0, ..., t_N$

The algorithm for simulating the VG process by DTGP bridge sampling (DTGP(B)) is presented in Algorithm 16.

Algorithm 16 *DTGP(B)*

1. *Initialization:*
   - *VG parameters:* θ, σ, ν
   - *Size of sample path* $N = 2^k$*, time to maturity* T
   - *Calculate the parameters of two gamma processes:* $\mu_p, \nu_p, \mu_n, \nu_n$ *using formula (4.6)*
   - $G_p(t_0) = 0$*,* $G_n(t_0) = 0$*,* $\Delta_t = T/N$
   - *Generate* $G_p(t_N) \sim \mathcal{G}(t_N \mu_p^2/\nu_p, \nu_p/\mu_p)$
   - *Generate* $G(t_N) \sim \mathcal{G}(t_N \mu_n^2/\nu_n, \nu_n/\mu_n)$

2. Simulation:

- *For* $l = 1$ *to* k
  - $m = 2^{k-l}$
  - *For* $j = 1$ *to* 2^{l-1}
    - $i = (2j-1)m$
    - $d = (t_i - t_{i-m})/\nu$
    - *Generate* $Y^+ \sim \mathcal{B}(d,d)$; $Y^- \sim \mathcal{B}(d,d)$
    - $G_p(t_i) = G_p(t_{i-m}) + [G_p(t_{i+m} - G_p(t_i)]Y^+$
    - $G_n(t_i) = G_n(t_{i-m}) + [G_n(t_{i+m} - G_n(t_i)]Y^-$
    - $X(t_i) = G_p(t_i) - G_n(y_i)$

3. Output: (X_t), $t = t_0, ..., t_N$

4.3 The variance gamma model in option pricing

The stock price process for the VG model is:

$$S(t) = S(0)\exp((m + \omega_S)t + X(t; \nu_S, \theta_S, \sigma_S)) \tag{4.9}$$

where m is the mean rate of return of the stock under the statistical probability measure, and the constant ω_S is chosen so that there is no arbitrage. The process $X(t; \nu_S, \theta_S, \sigma_S)$ is the VG process. By the Fundamental Theorem of Asset Pricing [17], the discounted asset price, $e^{-mt}S(t)$, must be a martingale, which requires $E[e^{-mt}S(t)] = S(0)$, or equivalently $E[e^{X(t)}] = e^{-\omega_S t}$. Thus, we have

$$\omega_S = -\ln(\phi_{X(t)}(-i)) = \frac{\ln(1 - \theta_S\nu_S - \sigma_S^2\nu_S/2)}{\nu_S},$$

and the subscripts "S" are used to express that the parameters are the statistical parameters.

Note that there is no unique martingale measure for the VG process due to the pure jump component. Gerber and Shiu [25] used the Esscher transform to find an equivalent martingale measure of the discounted stock prices of stochastic processes with stationary and independent increments. Yoo [66] used this method to find an equivalent martingale measure of the discounted stock prices for the VG process. Deville [19] used a method called mean-correcting martingale measure to obtain an equivalent martingale measure.

Under the risk neutral process, stock price process discounted at the risk free interest rate is a martingale and so the mean rate of return on the stock under the risk neutral probability measure

is the continuously compounded risk free interest rate r. The risk neutral process with no dividends and constant risk-free interest rate, r, is given by

$$S(t) = S(0)\exp((r+\omega_{RN})t + X(t;\sigma_{RN},\nu_{RN},\theta_{RN})) \tag{4.10}$$

where the constant ω_{RN} is chosen so that the discounted stock price is a martingale, i.e.,

$$\omega_{RN} = \frac{\ln(1-\theta_{RN}\nu_{RN}-\sigma_{RN}^2\nu_{RN}/2)}{\nu_{RN}}.$$

$X(t;\sigma_{RN},\nu_{RN},\theta_{RN})$ is the variance gamma process with parameters: $\sigma_{RN},\nu_{RN},\theta_{RN}$. For convenience, under the risk neutral process, we omit the subscripts "RN", and write the risk neutral process as

$$S(t) = S(0)\exp((r+\omega)t + X(t;\sigma,\nu,\theta)). \tag{4.11}$$

The condition for the existence of the martingale is

$$\omega = \frac{\ln(1-\theta\nu-\sigma^2\nu/2)}{\nu}.$$

Therefore the VG parameters must satisfy the condition:

$$\theta\nu - \sigma^2\nu/2 < 1.$$

The risk neutral process with dividend, d, can be written as

$$S(t) = S(0)\exp((r-d+\omega)t + X(t;\sigma,\nu,\theta)). \tag{4.12}$$

4.4 MC and QMC methods in option pricing

Suppose the payoff function of an option is $H_T = H_T(\omega)$. The price of the option at time $t < T$ is

$$P_t = E[H_T e^{-r(T-t)}]. \tag{4.13}$$

The price of the option in the VG model can be approximated by constructing a set $\{\hat{\omega}^m\}_{m=1,\ldots,M}$ of discrete sample paths randomly generated using the VG process. Then the approximation $\hat{P}_t$ of P_t is

$$\hat{P}_t = e^{-r(T-t)}\frac{1}{M}\sum_{m=1}^{M} H_T(\hat{\omega}^m). \tag{4.14}$$

For each discrete sample path for the VG process $X(t)$ over the period $[0,T]$, we use N time steps $0 = t_0 < t_1 < \ldots < t_N = T$. The MC algorithm for calculating the option price in the VG model is:

Algorithm 17 *European call option pricing in VG*

1. *Input VG parameters:* θ, σ, ν*; number of time steps used in the discretization* N*, time to maturity* T*, and initial stock price* S_0*;*
2. *For* $i = 1$ *to* M
   (a) *Simulate the VG processes* $\{X(t)\}_{t=t_0,...,t_N}$
   (b) *Simulate the stock price process,* $\omega_i = \{S(t)\}_{t=t_0,...,t_N}$*, using the formula (4.12)*
   (c) *Calculate the payoff* $H_T(\omega_i)$
3. *Calculate the option price:* $\hat{P}_0 = e^{-rT}\frac{1}{M}\Sigma_{i=1}^{M} H_T(\omega_i)$.

For the European call option, the payoff function is $H_T(\omega) = \max(S(T) - K, 0)$, where K is the strike price. In the Monte Carlo method, we use pseudorandom numbers for generating the sample paths. In the quasi-Monte Carlo method, we use the random start Halton sequence for sampling.

4.5 Bounds on the stock price process

In this section, we present bounds for the underlying stock prices, when they are obtained by the VG process given as the difference of two gamma processes. Let $\zeta = \omega + r - d$, $\zeta^+ = \max(\zeta, 0)$, $\zeta_- = \max(-\zeta, 0)$, then the stock price process in (4.12) can be written as

$$S(t) = S(0)\exp[(\zeta t + X(t)] = S(0)\exp[\zeta t + G_p(t) - G_n(t)]. \tag{4.15}$$

Define the upper, U_m, and lower, L_m processes over $[0, T]$ by

$$\begin{aligned} U_m(t) &= S(0)\exp[\zeta t + G_p(t_{m,i}) - G_n(t_{m,i-1}] \\ &= S(t_{m,i-1})\exp[\zeta(t - t_{m,i-1}) + G_p(t_{m,i}) - G_n(t_{m,i-1})] \end{aligned} \tag{4.16}$$

and

$$\begin{aligned} L_m(t) &= S(0)\exp[\zeta t - G_n(t_{m,i}) + G_p(t_{m,i-1}] \\ &= S(t_{m,i-1})\exp[\zeta(t - t_{m,i-1}) - G_n(t_{m,i}) + G_n(t_{m,i-1})] \end{aligned} \tag{4.17}$$

for $t_{m,i-1} < t < t_{m,i}$, and $L_m(t_{m,i}) = U_m(t_{m,i}) = S(t_{m,i})$, for $i = 0, ..., m$. L_m and U_m are both left and right discontinuous at the observation time $t_{m,i}$.

Proposition 1 *For every any path, integer* $m > 0$*, and all* $t \in [0, T]$*, we have*

$$L_m(t) \le L_{m+1}(t) \le S(t) \le U_{m+1}(t) \le U_m(t). \tag{4.18}$$

Avramidis and L'Ecuyer [2] stated this proposition and proved it. In the following, we present a simpler proof that utilizes the definitions of $L_m(t)$, $U_m(t)$ and the properties of gamma process.

Proof

At the observation time $t_{m,i}$, for $0 \leq i \leq m$, (4.18) holds by definitions of L_m and U_m. For $t_{m,i-1} < t < t_{m,i}$, for $0 \leq i \leq m$, we have:

$$\begin{aligned} L_m(t) &= S(0)\exp[\zeta t - G_p(t_{m,i}) + G_p(t_{m,i-1}] \\ &< S(0)\exp[\zeta t - G_p(t) + G_n(t)] = S(t) \end{aligned} \tag{4.19}$$

since $G_p(t)$ and $G_n(t)$ are increasing gamma functions and $t_{m,i-1} < t < t_{m,i}$. Similarly, we have $S(t) < U_m(t)$, therefore, for any integer $m > 0$, we have:

$$L_{m+1}(t) \leq S(t) \leq U_{m+1}(t).$$

Now we prove that $L_m(t) \leq L_{m+1}(t)$ and $U_{m+1}(t) \leq U_m(t)$. Recall that by the DTGP(B) algorithm, at step $m+1$ we generate S at the observation time y_{m+1} conditional on S at the observation times $\{y_0, y_1, ..., y_m\}$ or $\{t_{m,0}, t_{m,1}, ..., t_{m,m}\}$. Let $(t_{m,j-1}, t_{m,j})$ be the interval that contains the point y_{m+1}. Then for $0 \leq i \leq j-1$, $t_{m+1,i} = t_{m,i}$, and for $j \leq i \leq m$, $t_{m+1,i+1} = t_{m,i}$. Then we only need to prove $L_m(t) \leq L_{m+1}(t)$ for $t \in (t_{m,j-1}, t_{m,j}) \equiv (t_{m+1,j-1}, t_{m+1,j+1})$. Write $(t_{m+1,j-1}, t_{m+1,j+1}) = (t_{m+1,j-1}), t_{m+1,j}) \cup \{t_{m+1,j}\} \cup (t_{m+1,j}, t_{m+1,j+1})$. For $t = t_{m+1,j}$, $L_m(t) = L_{m+1}(t) = S(t)$ by definition. For $t_{m+1,j-1} < t < t_{m+1,j}$ we have

$$\begin{aligned} L_{m+1}(t) &= S(t_{m+1,j-1})\exp[\zeta(t - t_{m+1,j-1}) - G_n(t_{m+1,j}) + G_n(t_{m+1,j-1})] \\ &\geq S(t_{m+1,j-1})\exp[\zeta(t - t_{m+1,j-1}) - G_n(t_{m+1,j+1}) + G_n(t_{m+1,j-1})] \\ &= S(t_{m,j-1})\exp[\zeta(t - t_{m,j-1}) - G_n(t_{m,j}) + G_n(t_{m,j-1})] \\ &= L_m(t) \end{aligned}$$

because $t_{m+1,j-1} = t_{m,j-1}$, $t_{m+1,j+1} = t_{m,j}$, and $G_n(t_{m+1,j}) < G_n(t_{m+1,j+1})$. For $t \in (t_{m+1,j}, t_{m+1,j+1})$ we have

$$\begin{aligned} L_{m+1}(t) &= S(0)\exp[\zeta t - G_n(t_{m+1,j+1}) + G_p(t_{m+1,j})] \\ &\geq S(0)\exp[\zeta t - G_n(t_{m+1,j+1}) + G_p(t_{m+1,j-1})] \\ &= S(0)\exp[\zeta t - G_n(t_{m,j}) + G_p(t_{m,j-1})] \\ &= L_m(t) \end{aligned}$$

since $t_{m+1,j-1} = t_{m,j-1}$, $t_{m+1,j+1} = t_{m,j}$, and $G_p(t_{m+1,j}) > G_p(t_{m+1,j-1})$. So $L_m(t) \leq L_{m+1}(t)$ for any $t \in [0, T]$. Similarly, we have $U_{m+1}(t) < U_m(t)$, thus (4.18) holds for any positive integer m and $t \in [0, T]$. This completes the proof.

Define

$$\begin{aligned} L_{m,i} &= \inf_{t_{m,i-1} < t < t_{m,i}} L_m(t) \\ &= S(t_{m,i-1})\exp[-\zeta^-(t_{m,i} - t_{m,i-1}) - G_n(t_{m,i}) + G_n(t_{m,i-1})], \end{aligned}$$

$$\begin{aligned} U_{m,i} &= \sup_{t_{m,i-1}<t<t_{m,i}} U_m(t) \\ &= S(t_{m,i-1}) \exp[\zeta^+(t_{m,i} - t_{m,i-1}) + G_p(t_{m,i}) - G_p(t_{m,i-1})], \end{aligned}$$

$L^*_m(t) = L_{m,i}$ and $U^*_m(t) = U_{m,i}$ for $t_{m,i-1} < t < t_{m,i}$, and $L^*_m(t_{m,i}) = U^*_m(t_{m,i}) = S(t_{m,i})$, for $i = 1, ..., m$.

Corollary 4 *For every sample path S, any integer $m > 0$, and all $t \in [0, T]$, we have*

$$L^*_m(t) \le L^*_{m+1}(t) \le S(t) \le U^*_{m+1}(t) \le U^*_m(t).$$

4.6 Bounds on the option price

For each positive integer m, define

$$\Im_m = (t_{m,1}, G_p(t_{m,1}), G_n(t_{m.1}), ..., t_{m,m}, G_p(t_{m,m}), G_n(t_{m,m})), \tag{4.20}$$

and $C_{L,m}, C_{U,m}, C^*_{L,m}, C^*_{U,m}$ are the discounted payoff functions corresponding to the upper and lower bounds L_m, U_m, L^*_m, U^*_m, respectively.

Corollary 5 *Suppose that conditional on $\Im_m$, the payoff, C, is a monotone non-decreasing function of $S(t)$ for all values of t not in $\{t_{m,0}, t_{m,1}, ..., t_{m,m}\}$. Then*

$$C^*_{L,m} \le C_{L,m} \le C \le C_{U,m} \le C^*_{U,m}.$$

If C is (conditionally) non-increasing instead, the reverse inequality holds. In both cases, $|C^*_{U,m} - C^*_{L,m}| \to 0$ as $m \to \infty$.

Proof

The proof follows from Proposition 1 and the monotonicity of the function C.

4.7 Results

The VG process can be simulated by sequential sampling and bridge sampling. We presented these generation algorithms in Section 4.2. These algorithms utilize the normal, gamma, and beta distributions. The price of an European call option depends on the stock's price at the maturity only. Therefore, we only need to use one time step to simulate the VG process to estimate European call options. However, for path dependent option prices, such as Asian or American options, more

than one time step is required in the simulations. In our numerical results, we use both one time step and four time steps to calculate European call options. This allows us to use each of the four algorithms presented in Section 4.2 to simulate VG processes and compare the results. The parameters of the VG model are taken from an example in [58]. The results of using one time step and four time steps are presented in Tables 4.1 and 4.2, respectively.

In Table 4.1 we used sequential sampling and the Gamma time-changed Brownian motion algorithm, TCBM(S) (Algorithm 13), to generate the VG process using one time step. In this table, we report the price of a European call option with various maturities (given in the first column), when the underlying process is VG. The generation of the VG process with these particular parameters requires the generation of the following distributions: $\mathcal{G}(0.83, 0.3)$, $\mathcal{G}(1.67, 0.3)$, $\mathcal{G}(2.5, 0.3)$, and $\mathcal{G}(3.33, 0.3)$ corresponding to $T = 0.25, 0.5, 0.75, 1.0$, as well as the normal distribution. Consequently, our acceptance-rejection based approach to simulate the VG process uses Algorithms 6, 11, and 12. We call this approach **AR MC** and **AR QMC**, for the MC and QMC implementations of these algorithms. In Table 4.1, methods **Inverse MC** and **Inverse QMC** generate gamma and normal distributions using the inverse transformation method. In the **AR QMC** method, there are two cases. If the maturity is $T = 0.25$, then we generate a 4-dimensional (randomized) QMC vector (q_1, q_2, q_3, q_4). The first component is used to sample from the normal distribution using the inverse transformation method, and the last three components are used to sample from the gamma distribution using Algorithm 12. If $T > 0.25$, then we generate a 3-dimensional (randomized) QMC vector, use its first component to sample from the normal distribution by the inverse transformation method, and use the last two components to sample from the gamma distribution by Algorithm 11. In the **Inverse QMC** method, to obtain one option price, we generate a 2-dimensional (randomized) QMC vector (q_1, q_2). The first component is used to sample from the normal distribution, and the second component is used to sample from the gamma distribution. In Table 4.1 the second column "Exact price" reports the analytical value of the option price (see [39]). These values are taken from an example in [58].

In Table 4.2 we use four time steps to calculate the European call option and present the option prices at time $t = 0.5$ (year), using four Algorithms (13, 14, 15, 16) to simulate the VG process. For the sequential sampling algorithms (Algorithms 13 and 14) we need to generate variables from the normal distribution and gamma distribution; for the bridge sampling algorithms (Algorithms

15 and 16) we need to generate variables from normal distribution, gamma distribution, and beta distribution. The computed times recorded in Table 4.2 are the total times used for the four time steps simulations.

For each maturity, we compute the option price by generating 10,000 stock price paths. We then independently repeat this procedure 100 times. The sample standard deviation of the resulting 100 estimates is reported in Tables 4.1 and 4.2 (Std dev), together with the average of the 100 estimates (Price), and the total computing time.

Table 4.1: Comparison of inverse and acceptance-rejection methods in pricing European call options in the variance gamma model using one time step. The option parameters are: $\theta = -0.1436$, $\sigma = 0.12136$, $\nu = 0.3$, initial stock price $S_0 = 100$, strike price $K = 101$, and risk free interest rate $r = 0.1$.

Maturity	Exact		Inverse MC	AR MC	Inverse QMC	AR QMC
0.25	3.47	Std dev	0.039	0.035	0.002	0.003
		Price	3.47	3.47	3.47	3.47
		Time(s)	9.55	0.74	9.56	0.71
0.50	6.24	Std dev	0.062	0.063	0.002	0.005
		Price	6.23	6.25	6.24	6.24
		Time(s)	10.43	0.66	10.9	0.64
0.75	8.69	Std dev	0.079	0.093	0.003	0.007
		Price	8.68	8.70	8.69	8.69
		Time(s)	11.18	0.65	11.29	0.62
1.00	10.98	Std dev	0.106	0.103	0.003	0.009
		Price	10.99	10.99	10.98	10.98
		Time(s)	11.86	0.63	11.86	0.61

We make the following observations based on Tables 4.1 and 4.2:

1. The acceptance-rejection algorithms run faster than the inverse algorithms by factors between 9 and 25.

2. Both the acceptance-rejection method and the inverse method give results with sample standard deviations less than 0.01. These indicate that the estimation error of option prices would be within 1 cent.

Table 4.2: Comparison of inverse and acceptance-rejection methods in pricing European call options in the variance gamma model using four time steps. The option parameters are: $\theta = -0.1436$, $\sigma = 0.12136$, $\nu = 0.3$, initial stock price $S_0 = 100$, strike price $K = 101$, and risk free interest rate $r = 0.1$.

Algorithm	Price at t=0.5	Inverse MC	AR MC	Inverse QMC	AR QMC
TCBM(S)	Std dev	3.5e-3	3.6e-3	1.3e-5	6.9e-4
	Time(s)	38.16	2.99	38.11	2.90
DTGP(S)	Std dev	4.2e-3	3.1e-3	1.0e-5	8.3e-4
	Time(s)	92.24	3.87	90.79	3.68
TCBM(B)	Std dev	4.7e-3	3.6e-3	3.6e-4	4.8e-4
	Time(s)	17.99	2.50	17.87	2.47
DTGP(B)	Std dev	4.1e-3	2.7e-3	1.9e-5	8.4e-4
	Time(s)	60.93	2.98	60.08	2.96

CHAPTER 5

HIDDEN MARKOV MODELS

5.1 Introduction

In many real-world problems, the states of a system can be modeled as a Markov chain in which each state depends upon the previous state in a non-deterministic way. In hidden Markov models (HMMs), these states are invisible while observations (the inputs of the model), which depend on the states, are visible. An observation at time t of an HMM has a certain probability distribution corresponding with a possible state. The mathematical foundations of HMMs were developed by Baum and Petrie in 1966 [8]. Four years latter, in 1970, Baum and his colleagues published a maximization method in which the parameters of HMMs are calibrated using a single observation [9]. In 1983, Levinson, Rabiner and Sondhi [36] introduced a maximum likelihood estimation method for HMMs with multiple observation training, assuming that all the observations are independent. In 2000, Li et. al. [37] presented an HMM training for multiple observations without the assumption of independence of observations. HMMs have applications in many fields, including speech recognition, gene prediction, and machine translation. In this chapter, we will briefly introduce the concepts of an HMM and its applications in the prediction of the economics regimes and stock prices, using both single observation data and multiple observation data.

5.1.1 Elements of an hidden Markov model

We first introduce the basic elements of an hidden Markov model: $O = \{O_t, t = 1, 2, \ldots, T\}$ is the observation sequence; $Q = \{q_t, t = 1, 2, \ldots, T\}$ is the *hidden* state sequence; $\{S_i, i = 1, 2, \ldots, N\}$ are the possible values of each state q_t; N is the number of states that can be taken at a given time; M is the number of observation symbols per state, denoted by $V = \{v_k, k = 1, 2, \ldots, M\}$. The transition matrix A is defined as

$$A : a_{ij} = P(q_t = S_j | q_{t-1} = S_i),\ i, j = 1, 2, \ldots, N.$$

Here, we consider A as time-independent. The initial probability of being in state (regime) S_i is given by

$$p : p_i = P(q_1 = S_i),\ i = 1, 2, \ldots, N.$$

The observation probability is given by the matrix $B = (b_i(k))$, where

$$b_i(k) = P(O_t = v_k | q_t = S_i),\ i = 1, 2, ..., N;\ k = 1, 2, \ldots, M.$$

If the observation probability assumes the Gaussian distribution then $b_i(O_t) = \mathcal{N}(O_t, \mu_i, \sigma_i)$, where μ_i and σ_i are the mean and variance of the normal distribution corresponding to the state S_i, respectively. In summary, the parameters of an HMM are A, B, and p. For convenience, we use a compact notation for the parameters, given by

$$\lambda \equiv \{A, B, p\}.$$

If the observations are distributed by the Gaussian distributions, the parameters are

$$\lambda \equiv \{A, \mu, \sigma\, p\},$$

where μ and σ are vectors of means and variances of the Gaussian distributions, respectively.

5.1.2 Three problems

In order to apply hidden Markov models in real-world problems, there are three main problems that must be solved. They are described as follows:

1. Given the observation data $O = \{O_t, t = 1, 2, \ldots, T\}$ and the model parameters $\lambda = \{A, B, p\}$, how do we compute the probability of observations, $P(O|\lambda)$?
2. Given the observation data $O = \{O_t, t = 1, 2, \ldots, T\}$ and the model parameters $\lambda = \{A, B, p\}$, how do we choose the best corresponding state sequence $Q = \{q_1, q_2, ..., q_T\}$?
3. Given the observation data $O = \{O_t, t = 1, 2, \ldots, T\}$, how do we calibrate HMM parameters, $\lambda = \{A, B, p\}$, to maximize $P(O|\lambda)$?

These problems can be efficiently solved using the algorithms that will be described in Section 5.2.

5.2 Algorithms

In this section, we introduce algorithms known as the forward algorithm, backward algorithm, Viterbi algorithm, and Baum-Welch algorithm. Either forward or backward algorithms [6, 7] can be used for problem one while both of these algorithms are used in the Baum-Welch algorithm for problem three. The Viterbi algorithm solves the problem two.

5.2.1 Forward algorithm

We define the joint probability function as $\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = S_i|\lambda)$. We can calculate $\alpha_t(i)$ recursively. The probability of observation $P(O|\lambda)$ is just the sum of the $\alpha_T(i)'s$.

Algorithm 18 *The forward algorithm*

1. *Initialization: for i=1,2,..., N*
$$\alpha_{t=1}(i) = p_i b_i(O_1).$$
2. *Recursion: for* $t = 2, 3, \ldots, T$, *and for* $j = 1, 2, \ldots, N$, *compute*
$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(O_t).$$
3. *Output:*
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

5.2.2 Backward Algorithm

Similar to the forward algorithm, we define the conditional probability $\beta_t(i) = P(O_{t+1}, O_{t+2}, .., O_T|q_t = S_i, \lambda)$, for $i = 1, ..., N$. Then we have the following recursive backward algorithm.

Algorithm 19 *The backward algorithm*

1. *Initialization: for* $i = 1, ..., N$
$$\beta_T(i) = 1.$$
2. *Recursion: for* $t = T-1, T-2, ..., 1$, *for* $i = 1, ..., N$
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j).$$
3. *Output:*
$$P(O|\lambda) = \sum_{i=1}^{N} p_i b_i(O_1) \beta_1(i).$$

5.2.3 The Viterbi algorithm

The Viterbi algorithm ([23], [61]) is used to solve the second problem of HMMs. The goal here is to find the best sequence of states Q^* when (O, λ) is given. While problem one has exactly one solution, this problem has many possible solutions. Among these solutions, we need to find the one with the "best fit". We define:

$$\delta_t(i) = \max_{q_1,2,\ldots,q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, \ldots, O_t|\lambda).$$

By induction we have

$$\delta_{t+1}(j) = \max_i[\delta_t(i)a_{ij}]b_j(O_{t+1}).$$

Using $\delta_t(i)$ we can solve for the most likely state q_t, at time t, as

$$q_t = \text{argmax}_{1\leq i\leq N}[\delta_t(i)],\ 1 \leq t \leq T.$$

The Viterbi algorithm is given below.

Algorithm 20 *Viterbi algorithm*

1. *Initialization:*

$$\delta_1(j) = p_j b_j(O_1), \quad j = 1, 2, ..., N;$$

$$\phi_1(j) = 0.$$

2. *Recursion: for* $2 \leq t \leq T$, *and* $1 \leq j \leq N$

$$\delta_t(j) \quad = \max_i[\delta_{t-1}(i)a_{ij}]b_j(O_{t+1})$$

$$\phi_t(j) \quad = arg\max_i[\delta_{t1}(i)a_{ij}]$$

3. *Output:*

$$q_T^* \quad = \text{argmax}_i[\delta_T(i)]$$

$$q_t^* \quad = \phi_{t+1}(q_{t+1}^*),\ t = T-1, \ldots, 1$$

The forward algorithm, backward algorithm and Viterbi algorithm can be used for multiple observation data with minor changes. We present the most important algorithm, the Baum-Welch algorithm, for two cases: single observation and multiple independent observations.

5.2.4 Baum-Welch algorithm

We turn to the solution for the third problem, which is the most difficult problem of HMMs where we have to find the parameters $\lambda = \{A, \mu, \sigma\, p\}$ to maximize the probability $P(O, \lambda)$ of an observation data $O = \{O_1, O_2, \ldots, O_T\}$. Unfortunately, given an observation data, there is no way to find the global maximized of $P(O, \lambda)$. However, we can choose the parameters such that $P(O|\lambda)$ is locally maximized using the Baum-Welch iterative method [18]. In order to describe the procedure, we define $\gamma_t(i)$, the probability of being in state S_i at time t, as:

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O, \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}.$$

The probability of being in state S_i at time t and state S_j at time $t+1$, $\xi_t(i, j)$ is defined as:

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O, \lambda)}.$$

Clearly,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j).$$

Algorithm 21 *Baum-Welch algorithm for one observation [56]*

1. *Initialization: input parameters λ, the tolerance tol, and a real number $\triangle$*
2. *Repeat until $\triangle < tol$*
   - *Calculate $P(O, \lambda)$ using forward algorithm 18*
   - *Calculate new parameters λ^*: for $1 \leq i \leq N$*

$$p_i^* = \gamma_1(i)$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)},\ 1 \leq j \leq N$$

$$b_i^*(k) = \frac{\sum_{t=1}^{T} |_{O_t = v_k} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)},\ 1 \leq k \leq M$$

   - *Calculate $\triangle = |P(O, \lambda^*) - P(O, \lambda)|$*
   - *Update $\lambda = \lambda^*$*
3. *Output: parameters λ.*

If the observation probability $b_i^*(k)$, defined in Section 5.1.1, is Gaussian, the update parameters are:

$$\mu_i^* = \frac{\sum_{t=1}^{T-1} \gamma_t(i) O_t}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\sigma_i^* = \frac{\sum_{t=1}^{T} \gamma_t(i)(O_t - \mu_i)(O_t - \mu_i)'}{\sum_{t=1}^{T} \gamma_t(i)}.$$

Algorithm 22 *Baum-Welch for L independent observations* $O = (O^{(1)}, O^{(2)}, ..., O^{(L)})$ *with maturities* $T = (T^{(1)}, T^{(2)}, ..., T^{(L)})$ *[37]*

1. *Initialization: input parameters* λ*, the tolerance tol, and a real number* $\triangle$
2. *Repeat until* $\triangle < tol$
   - *Calculate* $P(O, \lambda) = \Pi_{l=1}^{L} P(O^{(l)}|\lambda)$ *using the forward algorithm (18)*
   - *Calculate new parameters* $\lambda^* = \{A^*, B^*, p^*\}$*, for* $1 \leq i \leq N$

$$p_i^* = \frac{1}{L} \sum_{l=1}^{L} \gamma_1^{(l)}(i)$$

$$a_{ij}^* = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \xi_t^{(l)}(i,j)}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \gamma_t^{(l)}(i)}, \ 1 \leq j \leq N$$

$$b_i^*(k) = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} |_{O_t^{(l)} = v_k^{(l)}} \gamma_t^{(l)}(i)}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \gamma_t^{(l)}(i)}, \ 1 \leq k \leq M$$

   - *Calculate* $\triangle = |P(O, \lambda^*) - P(O, \lambda)|$
   - *Update*

$$\lambda = \lambda^*.$$

3. *Output: parameters* λ*.*

If the observation probability $b_i^*(k)$, defined in Section 5.1.1, is Gaussian, the update parameters are:

$$\mu_i^* = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \gamma_t^{(l)}(i) O_t^{(l)}}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}-1} \gamma_t^{(l)}(i)}$$

$$\sigma_i^* = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \gamma_t^{(l)}(i)(O_t - \mu_i^{(l)})(O_t^{(l)} - \mu_i^{(l)})'}{\sum_{l=1}^{L} \sum_{t=1}^{T^{(l)}} \gamma_t^{(l)}(i)}.$$

5.3 Using HMMs to Predict Economics Regimes

In this section we will discuss how to use HMMs to predict economics regimes. We will use both HMMs for single observation data, [56] using Algorithm 21, and multiple observation data, [37] using Algorithm 22, to predict economics trends. We use two regimes and assume that the distributions corresponding to the two regimes are two normal distributions with means and variances $(\mu, \sigma) = (\mu_i, \sigma_i), i = 1, 2$. Regime 1 represents the Bull market and regime 2 represents the Bear market. We assume regime 2 is the regime with lower $\mu_i/\sigma_i, i = 1, 2$. First, we need to choose the appropriate economics indicators for the model. There are several economics indicators that we will choose for this work: Inflation (CPI), Credit Index, Yield Curve, Commodity, and Dow Jones Industrial Average. We use monthly percentage changes from February 1947 through June 2013 of the economics indicators. Second, we use the fixed period of historical data (48 months) of the economics indicators above to calibrate the model parameters $\lambda = \{A, \mu, \sigma, p\}$ using the Baum-Welch algorithm [56], [37]. Then the final step is to use the obtained parameters to predict the corresponding hidden regimes and predict the probabilities of upcoming regimes (Bull or Bear market). After each prediction, we update the data by dropping off the oldest month in the past and adding the most recent month to keep the same length of time periods. We also update parameters and repeat the prediction process. We call this method a "moving window" method based on the approach of by Zhang [67] , Rao and Hong [57].

We use HMMs to predict the probability of being in the Bear market for the US economics from October 2006 to May 2013 using both single observation and multiple observations. In Figure 5.1 we use the Dow Jones Industrial Average (DJIA) and in Figure 5.2, we use the inflation rate (CPI). Figures 5.1 and 5.2 show that HMM captures the economics crisis from the end of 2008 to 2009: the probabilities of being in the Bear market at that time are almost 1. We see that different economics indicators give different predictions for economics regimes at some points in the time period. However, both figures suggest that HMM can predict economics crisis using either the stock indicator or the inflation indicator. In Figure 5.3 we use HMM with multiple observation data to predict the economic regimes. We assume that the economics indicators are independent.

5.4 Using HMMs to Predict Stock Prices

In this section we will use HMMs to predict the daily stock prices. We predict one year daily stock prices, and we fix the length of the "moving window" to 252 days (one business year). The stock price prediction follows the three steps of economics regime prediction that was discussed in the previous section. However, in the final step, we need to do more work in order to predict stock prices. In the economics regime prediction, after training the model, we use the transition matrix to predict the probability of being in the Bear market. Now suppose we want to predict tomorrow's stock price. After training the model parameters we calculate the probability of observation for today and move the "moving window" back to the past, day by day, to find the day in the past that has the closest probability of observation data with today. Then we calculate the difference between that day to the day after. Thus tomorrow's stock closing price forecast is established by adding the above difference to today's closing price. This prediction approach is based on a paper of Hassan and Nath [29].

We present results of using HMMs to predict $S\&P500$ closing price for one year from July 2012 to July 2013 in Figures 5.4 and 5.5. In Figure 5.4, we use only the "close" price while in Figure 5.5, we use "close", "open", "low", and "high", to predict the closing prices. We compare the results by calculating the relative errors of the two estimations. The HMM using multiple observations has a relative error of 0.00703, which is smaller than the error of 0.00824 when using one observation data. The results show that HMMs can be used to predict stock prices; the predictions follow the trends of the actual prices. We then use the HMM algorithm in stock trading. We choose six stocks in the market to trade with equal weights: Google Inc. (GOOG), Ford Motor Company (F), Apple Inc. (AAPL), SPDR $S\&P500$ ETF Trust (SPY), and General Electric Company (GE). We use HMM for history prices of each stock to predict the stock prices for the year December 2012 to December 2013. If HMM predicts that the stock price will move up tomorrow, for example, we will buy today and sell tomorrow, assuming that we buy and sell with close prices. Table 5.1 shows that we make about 75% earnings after trading these six stocks for one year without transaction fees.

Table 5.1: One year daily stock trading portfolio from December 2012 to December 2013

Symbol	Initial Investment ($)	Earning ($)	Earning %
SPY	9,000.00	2050.66	22.79
GOOG	30,000.00	29,036.4	96.79
FORD	250.00	10.10	4.04
AAPL	950.00	19.06	2.01
GE	1,700.00	490.00	28.82
TOTAL	41,900.00	31,606.22	75.43

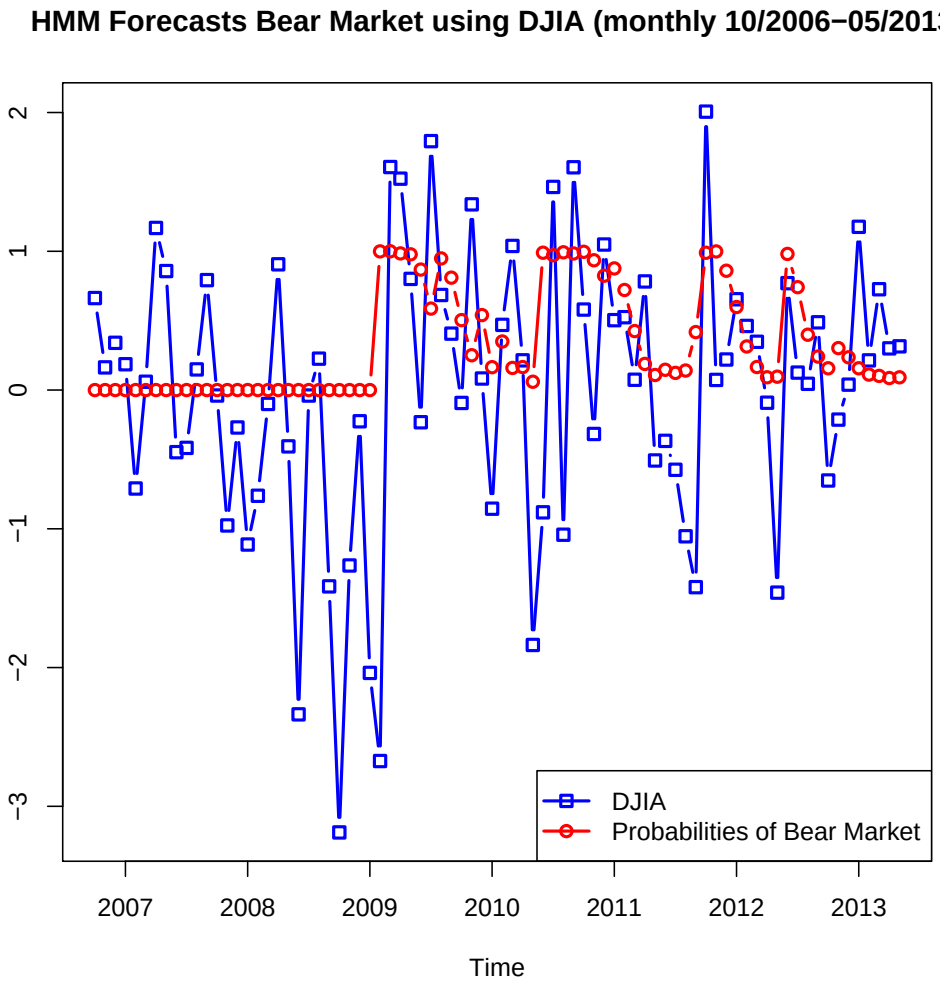

Figure 5.1: Forecast probabilities of being in the Bear market using DJIA indicator

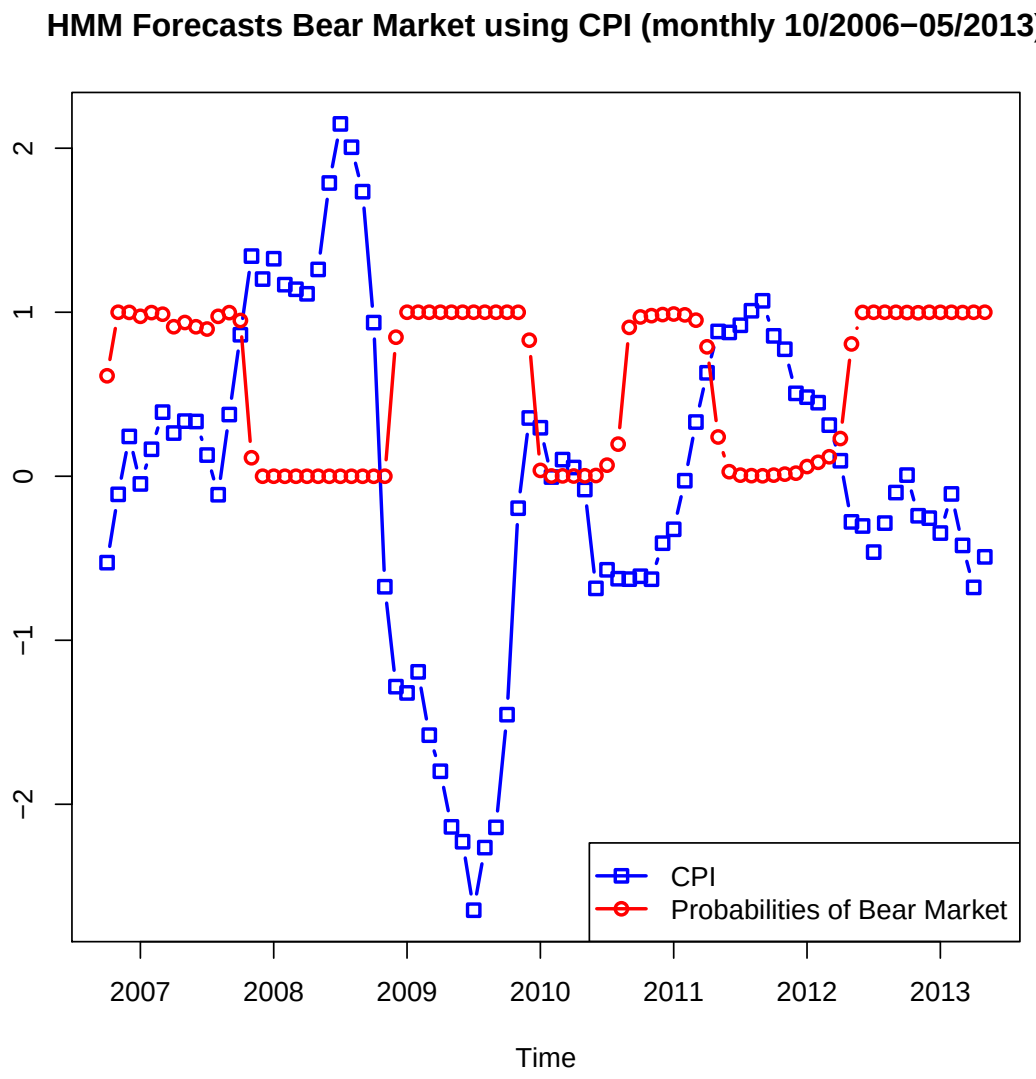

Figure 5.2: Forecast probabilities of being in the Bear market using CPI indicator

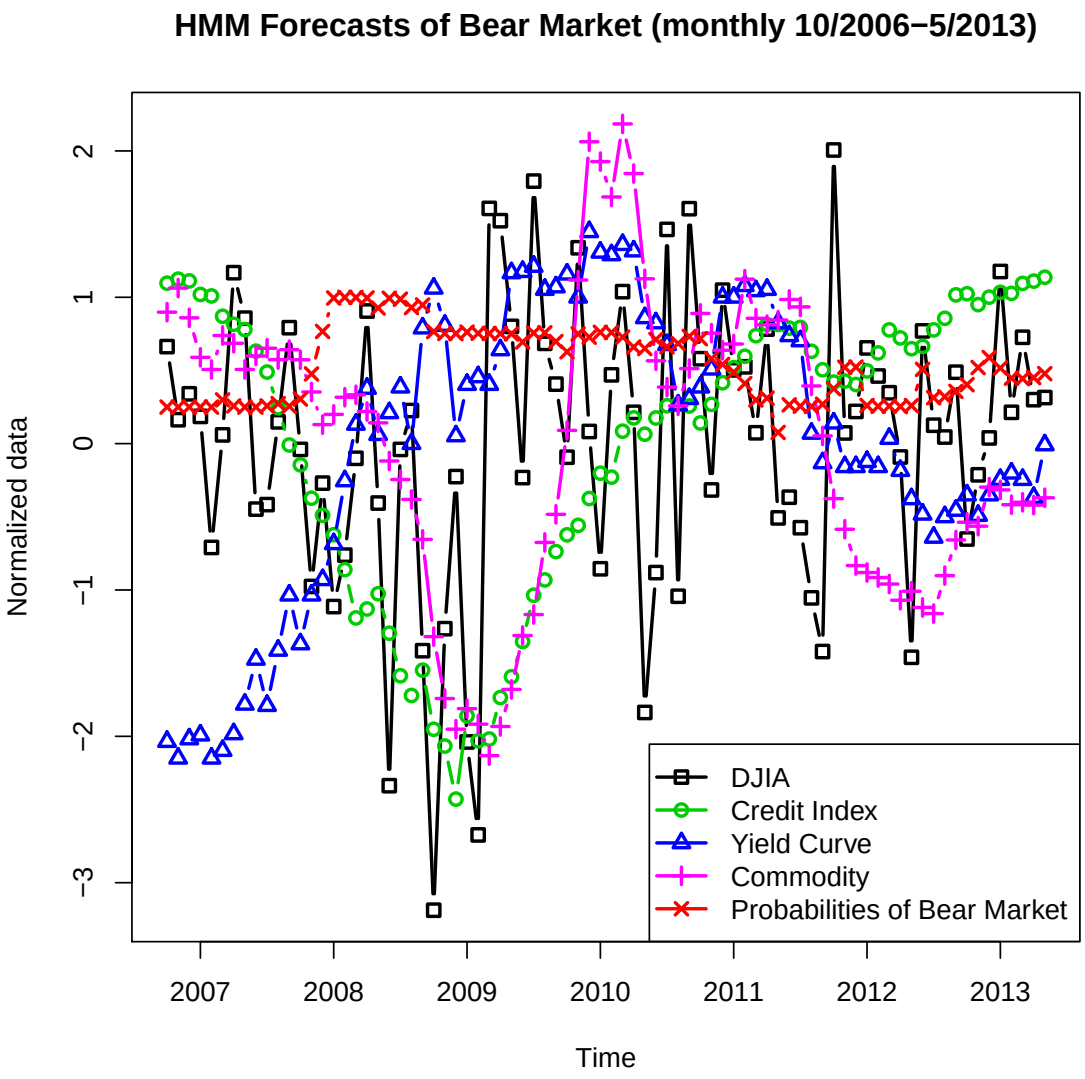

Figure 5.3: Forecast probabilities of being in the Bear market using multiple observations

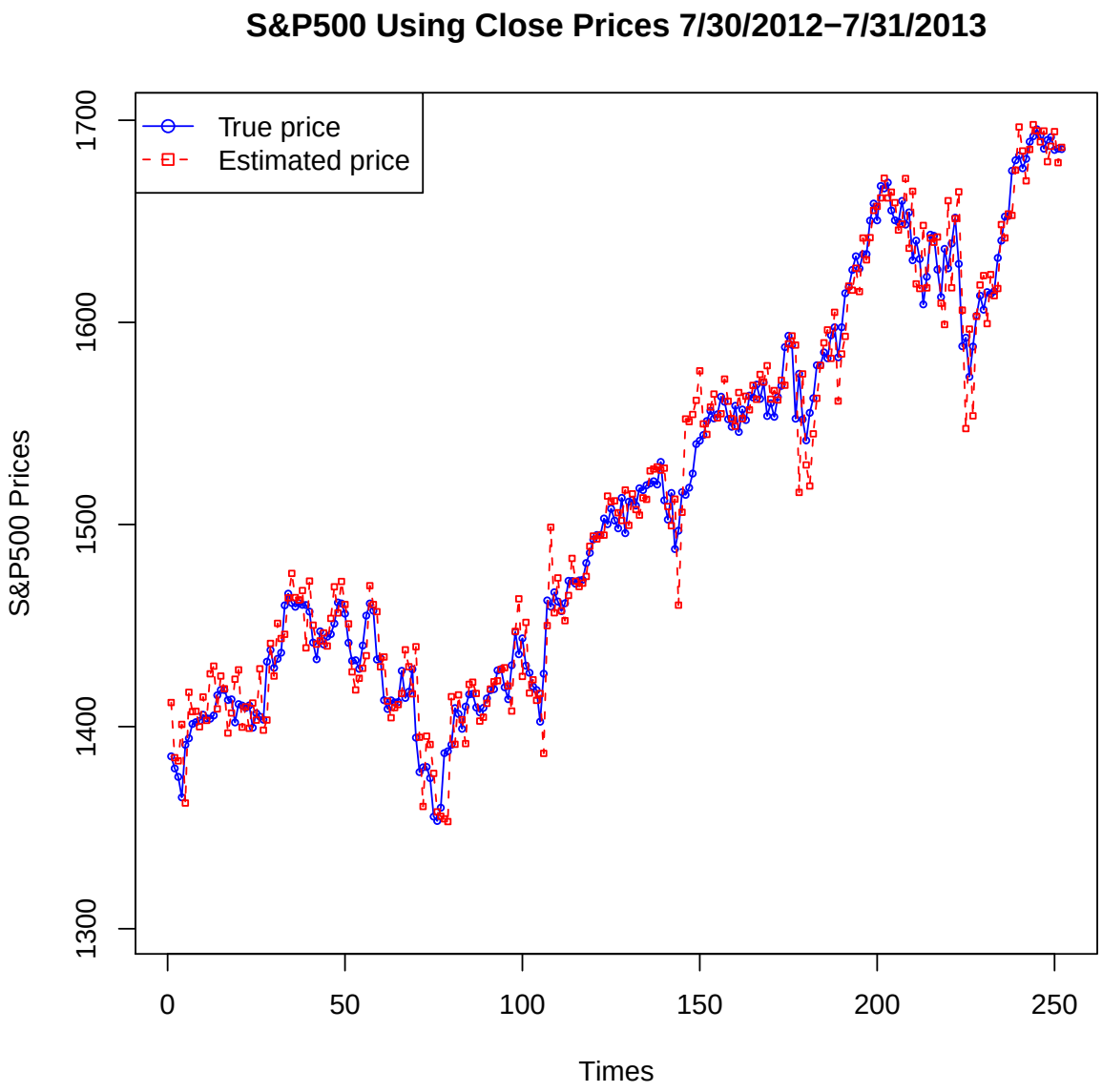

Figure 5.4: Forecast $S\&P500$ using "close" prices

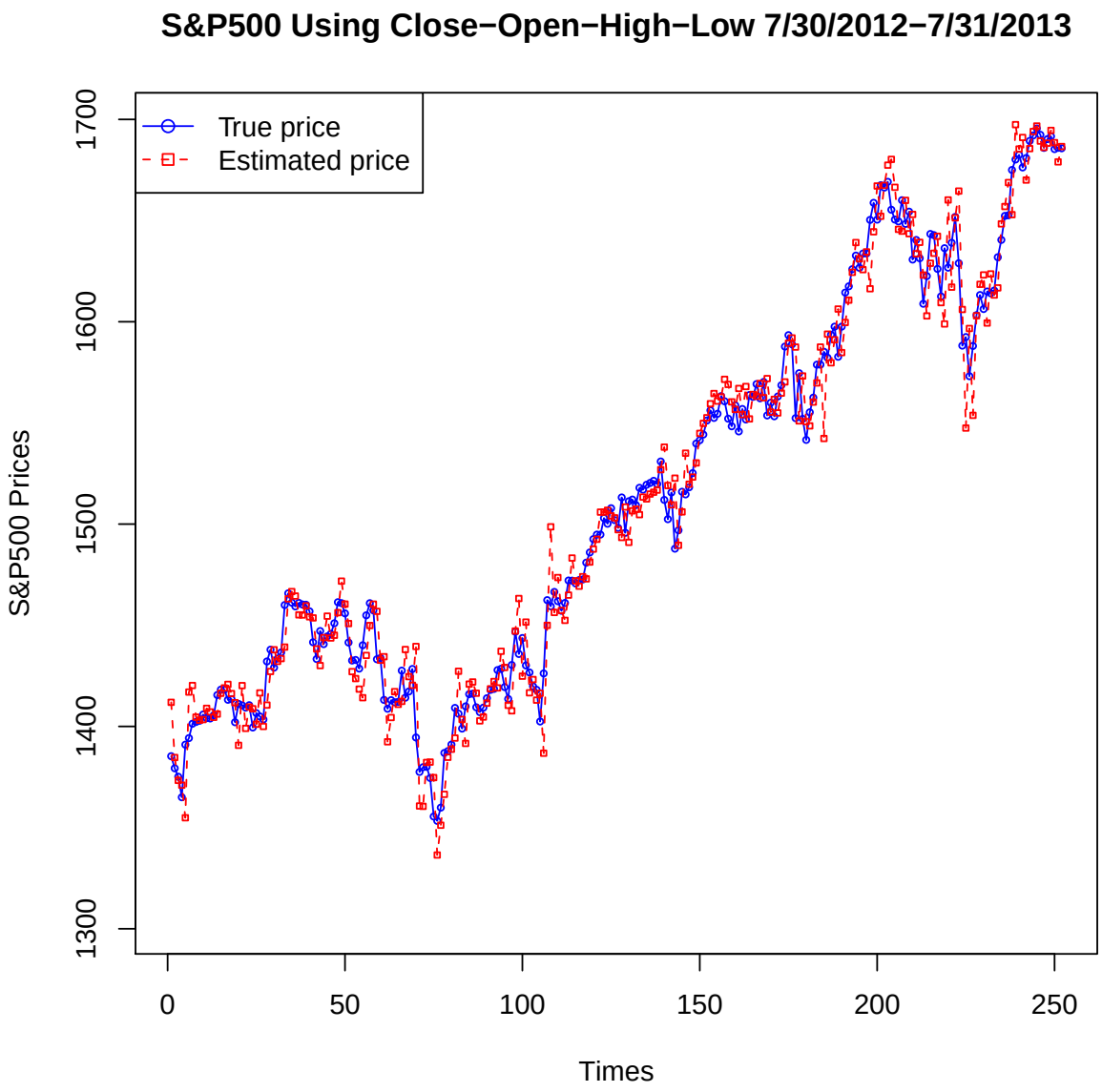

Figure 5.5: Forecast $S\&P500$ using "open", "close", "high", and "low" prices

CHAPTER 6

CONCLUSION

The use of low-discrepancy sequences in computational problems, especially in numerical integration, is increasing mainly because of the faster convergence rates these sequences provide, compared to using pseudorandom sequences. For example, in the application of derivative pricing from computational finance, this faster rate of convergence is quite useful, and some well-known low-discrepancy sequences have taken their place in the numerical methods toolbox of financial engineers. Currently, the main method for transforming low-discrepancy sequences to nonuniform distributions is the inverse transformation technique. However, this technique can be computationally expensive for complicated distributions. The acceptance-rejection technique was developed precisely for this reason for the pseudorandom sequences.

In this dissertation, we introduced an acceptance-rejection algorithm for the quasi-Monte Carlo method. We proved a convergence result for the algorithm and investigated its error bounds. We constructed several QMC acceptance-rejection algorithms to generate variables from different distributions. Our numerical results showed that the QMC acceptance-rejection method gave better results compared to the other methods such as ziggurat, smoothing, and inverse method, for the distributions considered in this dissertation. The availability of acceptance-rejection for low-discrepancy sequences significantly increases the scope of applications where quasi-Monte Carlo methods can improve traditional Monte Carlo.

We also investigated the applications of Hidden Markov Models (HMM) in predicting economic regimes and stock prices. We used HMM for both single and multiple observation data, assuming that the observations are independent. We used a "moving window" technique with a fixed time length period in predictions. Numerical results show that HMM using multiple observation data gave more accurate predictions than using single observation data only. To test our HMM stock prediction algorithm, we developed a simple trading strategy using the algorithm and obtained profits about 75% in one historical scenario. We also observed that HMM regime prediction algorithm was able to predict the economic crisis in 2008-2009.

BIBLIOGRAPHY

[1] E. Atanassov. On the discrepancy of the Halton sequences. *Math. Balkanica*, 18.1-2:15–32, 2004.

[2] A. N. Avramidis and P. L'Ecuyer. Efficient Monte Carlo and quasi-Monte Carlo option pricing under the Variance-Gammma model. *Management Science*, 52:1930–1934, 2006.

[3] A.N. Avramidis, P. L'Ecuyer, and P.A. Tremblay. Efficient simulation of gamma and variance-gamma processes. *Winter Simulation Conference*, 42:85–99, 1983.

[4] R.E. Caflisch B. Moskowitz. Smoothness and dimension reduction in quasi-Monte Carlo methods. *Math. Comput. Modelling*, 23:37–54, 1996.

[5] G. Bakshi, C. Cao, and Z. Chen. Empirical performance of alternative option pricing models. *Journal of Finance*, 52:20032049, 1997.

[6] L. E. Baum and J. A. Egon. An inequality with applications to statistical estiation for probabilistic functions of Markov process and to a model for ecology. *Bull. Amer. Meteorol. Soc*, 73:360–363, 1967.

[7] L. E. Baum and G. R. Sell. Growth functions for transformations on manifolds. *Pac. J. Math*, 27.2:211–227, 1968.

[8] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[9] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[10] D. Bayazit. Sensitivity analysis of options under Lévy processes via Malliavin calculus. *PhD thesis, Florida State University*, 2010.

[11] F. Black and M. Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3):637654, 1973.

[12] P. Carr and D. B. Madan. Option valuation using the fast fourier transform. *Journal of Computational Finance*, 1999.

[13] R.C.H. Cheng. The generation of gamma variables with non-integral shape parameter. *Applied Statistics*, 26(1):71–75, 1977.

[14] G.W. Cran, K.J. Martin, and G.E. Thomas. Remark AS R19 and Algorithm AS 109: a remark on Algorithms: AS 63: the incomplete beta integral AS 64: inverse of the incomplete beta function ratio. *J. R. Stat. Soc. Ser. C. Appl. Stat.*, 26(1):111–114, 1977.

[15] R.B. D'Agostino and M.A. Stephens. Goodness-of-fit techniques. *Marcel Dekker, New York*, 1986.

[16] G. Ökten and W. Eastman. Randomized quasi-Monte Carlo methods in pricing securities. *Journal of Economic Dynamics &Control, 28*, 2004.

[17] F. Delbaen and W. Schachermayer. A general version of the fundamental theorem of asset pricing. *Springer-Verlag*, 1994.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via EM algorithm. *J. Roy. Stat. Soc*, 39.1:1–38, 1977.

[19] D. Deville. On Lévy processes for option pricing : numerical methods and calibration to index options. PhD thesis, University of Politecnica, Delle Marche, 2008.

[20] H. Faure. Discrpance de suites associes un systme de numration (en dimension un). *Acta Arith*, 1982.

[21] H. Faure and C. Lemieux. Improvements on the star discrepancy of (t, s)-sequences. *Acta Arith.*, 151.1:61–78, 2012.

[22] G. S. Fishman. Monte Carlo concepts, algorithm and applications. *Springer Series in Operations Research*, 1996.

[23] G. D. Forney. The Viterbi algorithm. *IEEE*, 61:268–278, 1973.

[24] M. C. Fu. Variance-Gamma and Monte Carlo. Unpublished paper, available online at www.rhsmith.umd.edu/faculty/mfu/fu_files/Fu07.pdf, 2007.

[25] H. U. Gerber and E. S. W. Shiu. Option pricing by Essher transforms. *Transactions of Society of Actualizes*, 46, 1994.

[26] P. Glasserman. Monte Carlo methods in financial engineering. *Springer Science*, 2004.

[27] A. Göncü and G. Ökten. Generating low-discrepancy sequences from the normal distribution: Box-Muller or inverse transformation? *Math. Comput. Modelling, 53*, 2011.

[28] J.H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math*, 1960.

[29] Md. R. Hassan and B. Nath. Stock market forecasting using hidden Markov models: A new approach. *IEEE*, 2005.

[30] C. Joy, P. P. Boyle, and K. S. Tan. Quasi-Monte Carlo methods in numerical finance. *Management Science*, 1996.

[31] L. Kuipers and H. Niederreiter. Uniform distribution of sequences. *Dover Publications*, 2006.

[32] J.P. Lambert. Quasi-Monte Carlo, low discrepancy sequences, and ergodic transformations. *Journal of Computational and Applied Mathematics*, 1985.

[33] B. Lapeyre and G. Pagès. Families de suites d discrépance faible obtenues par itération de transformations de $[0, 1]$. *C.R. Acad. Sci. Paris Sr. I Math.*, pages 507–509, 1989.

[34] C.L. Lau. Algorithm AS 147: A simple series for the incomplete gamma integral. *Applied Statistics*, 29:113–114, 1980.

[35] P.H.W. Leong, G. Zhang, D-U Lee, W. Luk, and J.D. Villasenor. A comment on the implementation of the ziggurat method. *Journal of Statistical Software*, 12:1–4, 2005.

[36] S.E. Levinson, L.R. Rabiner, and M.M. Sondhi. An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition. *Bell System Technical Journal*, 62(4):1035–1074, 1983.

[37] X. Li, M. Parizeau, and R. Plamondon. Training hidden Markov models with multiple observations a combinatorial method. *IEEE Transactions on PAMI*, 22(4):371–377, 2000.

[38] D. B. Madan, P. Carr, and E. C. Chang. The variance gamma process and option pricing. *European Finance Review*, 2:79105, 1998.

[39] D. B. Madan and E. Seneta. The Variance-Gammma (VG) model for share market returns. *Journal of Business*, 63:511–524, 1990.

[40] K. L. Majumder and G.P. Bhattacharjee. Algorithm AS 63: The incomplete beta integral. *J. R. Stat. Soc. Ser. C. Appl. Stat.*, 22(3):409–411, 1973.

[41] G. Marsaglia. Generating a variable from the tail of the normal distribution. *Boeing Scientific Research Laboratories*, 1963.

[42] G. Marsaglia and W. W. Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software, Vol. 5, No. 8*, 2000.

[43] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulations*, 8(1):3–30, 1998.

[44] A. Mcleod. Algorithm AS 245: A robust and reliable algorithm for the logarithm of the gamma function. *J. R. Stat. Soc. Ser. C. Appl. Stat.*, 38(2):397–402, 1989.

[45] H. G. Meijer. The discrepancy of a adic sequence. *Indag. Math*, 1968.

[46] H. Niederreiter. Random number generation and quasi-Monte Carlo methods. *SIAM, Philadelphia*, 1992.

[47] H. Niederreiter. Error bound for quasi-Monte Carlo integration with uniform point sets. *Journal of Computational and Applied Mathematics*, 2003.

[48] H. Niederreiter and R. F. Tichy. Beiträge zur diskrepanz bezuglich gewichteter mittel. *Manuscripta Math.*

[49] J. R. Norris. Markov chains. *Cambridge University Press*, 1997.

[50] G. Ökten. Error reduction techniques in quasi-Monte Carlo integration. *Math. Comput. Modelling*, 30:61–69, 1999.

[51] G. Ökten. Generalized von Neumann-Kakutani transformation and random-start scrambled Halton sequences. *J. Complexity*, 25(4):318–331, 2009.

[52] G. Ökten and A. Göncü. Uniform point sets and the collision test. *J. Comput. Appl. Math.*, 259:798–804, 2013.

[53] A. B. Owen. Randomly permuted (t, m, s)-nets and (t, s)-sequences. *In H.Niederreiter and P.J.-S. Shiue, editors, Monte Carlo and Quasi-Monte Carlo in Scientic Computing, Lecture Notes in Statistics, Vol. 106*, 1995.

[54] A. B. Owen. Scrambled net variance for integrals of smooth functions. *Annals of Statistics*, pages 1541–1562, 1997.

[55] A. B. Owen. Multidimensional variation for quasi-Monte Carlo. *The World Scientific Publisher*, pages 49–74, 2005.

[56] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE*, 77.2, 1989.

[57] S. Rao and J. Hong. Analysis of hidden Markov models and support vector machines in financial applications. *University of California at Berkeley*, 2010.

[58] C. Ribeiro and N. Webber. Valuing path dependent options in the variance-gamma model by Monte Carlo with a variance gamma bridge. *working paper, WP02-04*, 2002.

[59] S. M. Ross. Simulation. *Academic Press*, 1997.

[60] J. Struckmeier. Fast generation of low-discrepancy sequences. *Journal of Computational and Applied Mathematics*, 61, 1995.

[61] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theory*, IT-13:260–269, 1967.

[62] J. Wang. The multivariate variance gamma process and its applications in multi asset option pricing. *PhD thesis, University of Maryland, College Park*, 2009.

[63] X. Wang. Improving the rejection sampling method in quasi-Monte Carlo methods. *Journal of Computational and Applied Mathematics 114*, 1999.

[64] X. Wang and F. J. Hickernell. Randomized Halton sequences. *Mathematical and Computer Modelling 32*, 2001.

[65] H. Wozniakowski. Average case complexity of multivariate integration. *Bull. Amer. Math. Soc*, 1991.

[66] E. J. Yoo. Variance gamma pricing of American futures options. *PhD thesis, Florida State University*, 2008.

[67] Y. Zhang. Prediction of financial time series with hidden Markov models. *MS thesis, Simon Fraser University*, 2004.

BIOGRAPHICAL SKETCH

Nguyet Nguyen

EDUCATION

- FLORIDA STATE UNIVERSITY — TALLAHASSEE, FL, USA
  Doctor of Philosophy, Financial Mathematics — *07/2014* (expected)
- FLORIDA STATE UNIVERSITY — TALLAHASSEE, FL, USA
  Master of Science, Financial Mathematics — *12/2011*
- HANOI NATIONAL UNIVERSITY OF EDUCATION — HANOI, VIETNAM
  Master of Science, Mathematics — *09/2002*
- HANOI NATIONAL UNIVERSITY OF EDUCATION — HANOI, VIETNAM
  Bachelor of Science, Mathematical Education — *06/1998*

WORKING EXPERIENCE

- FLORIDA STATE UNIVERSITY, DEPARTMENT OF MATHEMATICS, TALLAHASSEE, FL, USA
  Senior Teaching Assistant: *Spring 2013-Present*
  - Calculus with Analytic Geometry I — *Fall 2013*
  - Pre-Calculus Algebra — *Spring 2013* & *Spring 2014*

  Teaching Assistant:
  - Calculus with Analytic Geometry I — *Fall 2012*
  - Calculus II — *Spring 2012* & *Fall 2011*
  - College Algebra — *Fall 2010, Spring 2010,* & *Summer 2011*
- NGUYEN TAT THANH HIGH SCHOOL, HNUE — HANOI, VIETNAM
  Mathematics Teacher: *08/1998 - 12/2005*
  Director of the Division of Math Teachers: *2003-2005*
- NED DAVIS RESEARCH GROUP — VENICE, FL
  Quantitative Research Intern: *06/2013-08/2013*
  Quantitative Research Intern: *05/2012-08/2012*

OTHER TRAINING AND CERTIFICATIONS

- 2013 Program for Instructional Excellence (PIE) Certificate by the Graduate School of Florida State University TALLAHASSEE, FL, *09/ 2013*
- Official Statement of Status of Eligibility for a Florida Educators Certificate by the Florida Department of Education TALLAHASSEE, FL, *05/ 2007*
- Certification of Attendance of the Joint Training of Pedagogical Interactive by Universit*é* du Qu*é*bec *à* Trois-Rivières (Canada) and HNUE (Vietnam) HANOI, VIETNAM, *2005*
- Attended the Annual Training Courses in Child Psychology for Advisers for Children by UNICEF and Department of Psychology, HNUE HANOI, VIETNAM *2003* & *2004*

PUBLICATIONS

- Nguyet Nguyen and Giray Ökten, *Acceptance-rejection method for low-discrepancy sequences*, SIAM Journal on Scientific Computing, under review.

CONFERENCES

- The Eleventh International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, KU Leuven, Belgium, (accepted abstract) *04/2014*
  "Efficient Implementation of The Variance Gamma Model Using a QMC Acceptance-Rejection Algorithm"
- The 2014 Joint Mathematical Meeting, Baltimore, MD *01/2014*
  "Hidden Markov Model for High Frequency Data"
- The 5^{th} Annual Modeling High Frequency Data in Finance Conference, Hoboken, NJ *10/ 2013*
  "Hidden Markov Model for High Frequency Data"
- The American Mathematical Society Section Meeting, University of Louisville, Louisville, KY *10/2013*
- The Mathematical Association of America Florida Chapter Local Meeting, University of West Florida, Pensacola, FL *11/2012*
  "Applying Quasi-Monte Carlo for Acceptance-Rejection Algorithm"

HONORS AND AWARDS

- First place in the poster competition of the 15^{th} and 16^{th} annual Financial Math Festivals, Florida State University, FL *02/2013* and *02/2014*

- National Science Foundation (NSF) travel grant to attend the Eleventh International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing (MCQMC 2014), Leuven, Belgium *11/2013*
- American Mathematical Society (AMS) travel grant to attend the 2014 Joint Mathematical Meeting, Baltimore, MD *11/2013*
- Pi Mu Epsilon, National Honorary Mathematics society, Florida Beta Chapter: recognition of superior achievement in the field of mathematics. *Member since 2011*
- AMS travel grant to attend the AMS Section Meeting, University of Louisville, KY *10/2013*
- Douglas Peterson Vietnamese Scholarship Award, Florida State University, FL *09/2011* & *08/2013*
- Ermine M. Owenby Jr. Fund to Promote Excellence, Florida State University, FL *11/2012* & *11/2013*
- Teaching Assistant, Florida State University, USA *08/2010-Present*
- Annual fellowship for graduate students, Department of Mathematics, Hanoi National University of Education, Vietnam *1998-2002*
- Third prize in algebra, nationwide contest for undergraduate students of Vietnam *1996*
- Annual fellowship for excellent undergraduate students, Department of Mathematics, Hanoi National University of Education, Vietnam *1994-1998*