Valuing American Asian Options with Least Squares Monte Carlo and Low Discrepancy Sequences

by
A.J. van Niekerk

Submitted in partial fulfilment of the requirements for the degree
Master of Science in Financial Engineering

In the Faculty of Natural and Agricultural Sciences
Mathematics and Applied Mathematics Department
University of Pretoria
Pretoria
Nov 2018

I, AJ van Niekerk declare that the dissertation, which I hereby submit for the degree Master of Science in Financial Engineering at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

SIGNATURE:

DATE: Nov-2018

Abstract

There exists no closed form approximation for arithmetically calculated Asian options, but research has shown that closed form approximations are possible for Geometrically calculated Asian options. The aim of this dissertation is to effectively price American Asian options with the least squares Monte Carlo approach (Longstaff & Schwartz, 2001), applying Low discrepancy sequences and variance reduction techniques. We evaluate how these techniques affect the pricing of American options and American Asian options in terms of accuracy, computational efficiency, and computational time used to implement these techniques. We consider the effect of, Laguerre-, weighted Laguerre-, Hermite-, and Monomial-basis functions on the Longstaff and Schwartz (2001) model. We briefly investigate GPU optimization of the Longstaff and Schwartz algorithm within Matlab. We also graph the associated implied and Local volatility surfaces of the American Asian options to assist in the practical applicability of these options.

Keywords. Asian Options, Monte Carlo, Black-Scholes, Variance reduction, Control variates, Antithetic variates, Quasi random sequences, Longstaff and Schwartz, Halton sequence, Sobol sequence, least squares Monte Carlo, GPU optimization, Implied volatility surface, Local volatility surface

Contents

List of Abbreviations

A	Payoff of an Asian option
$C(S_t, X)$	Payoff of a Call option
Δ	Delta
LSM	Least squares Monte Carlo
μ	Drift term
$\mathcal{N}$	Cumulative normal distribution
Q	Martingale measure
r	Riskfree interest rate
S_0	Initial price of underlying
S_t	Price of the underlying at time t
σ	Volatility
T	Total time to expiration
t_i	Evenly spaced time increments
t_0	Current time
t	Some time between 0 and T
W_t	Stochastic term
X	Strike price

List of Definitions

List of Figures

List of Tables

Chapter 1

Introduction

An Asian option is an option where payoff depends on the average price of the underlying asset over a certain period as opposed to at maturity. It is also known as an average option [47] and is a path dependent style option. Asian options were first introduced in 1987 by the Bankers Trust in Tokoyo and their aim was to price these options on crude oil. Asian options are of importance and interest, to thinly traded assets, since both the investor and issuer of the Asian option enjoys some protection in very volatile market conditions [40]. Asian options are frequently priced on interest rates, foreign currencies, and also, more commonly known, commodities such as crude oil. There are two distinct Asian options that are calculated either by using the arithmetic or the geometric mean.

The calculation to obtain the average, which is used to price Asian options, could be calculated over the lifetime of the option, or over a fixed specified period. There exist some permutations of Asian options, called fixed- and floating- strike options. These options have the following payoff functions.
Fixed strike Asian options:

$$Asian\ call\ option = max[0, A(0,T) - X]$$

$$Asian\ put\ option = max[0, X - A(0,T)]$$

Floating strike Asian options:

$$Asian\ call\ option = max[0, S_T - kA(0,T)]$$

$$Asian\ put\ option = max[0, kA(0,T) - X]$$

where A denotes the average price for the period [0,T], and X is the strike price. S_T denotes the underlying assets price at time T. The k is defined as a weighting, but is usually set to 1. We can also define A as an arithmetic average for discrete and continuous cases, as follows.

$$A(0,T) = \frac{1}{T}\int_0^T S_t\ dt \qquad \text{(eq:1.0.1)}$$

$$A(0,T) = \frac{1}{n}\sum_{i=0}^{n-1} S(t_i) \qquad \text{(eq:1.0.2)}$$

For geometrically averaged Asian options, we express the calculation of the continuous case as

$$A(0,T) = exp(\frac{1}{T}\int_0^T ln(S_t)\ dt) \qquad \text{(eq:1.0.3)}$$

and for the discrete case,

$$A(0,T) = (\prod_{i=0}^{n-1} S(t_i))^{\frac{1}{n-1}} \qquad \text{(eq:1.0.4)}$$

where intervals are equidistant for the discrete case of time. For example $0 = t_0, t_1, t_2, ..., t_n = T$ and for any specific $t_i = i * \frac{T}{n}$. In special cases, where the average only depends on the average of the final price at time T, we obtain the normal European option.

The averaging period of an Asian option can span the lifetime of the option or any subset thereof. We will primarily be investigating the arithmetic Asian option, although the geometrically calculated Asian option plays a vital role in many pricing methods. Geometric Asian options enables us to find a closed form solution, as discussed in multiple papers such as Wilkmund [8], Hubalek and Sgarra [5], Jeon, Ji-Hun and Kang [7], and Kemna and Vorst [40]. Continuous cases of the Asian options are only valid in a theoretical context, because the industry relies heavily on the discrete calculation of the averages.

Due to the volatile nature of the mining sector of South Africa, discrete calculations of Asian options might be applicable. The South African mining industry has seen some volatile commodity price movements over the past years. According to PWC's [46] press room, "the market capitalization for the top 35 companies declined to R414 billion as at 30 June 2015 (compared to R675 billion as at 30 June 2014). The decline continued when compared to market capitalization as at 30 September 2015 of R304 billion, resulting in an aggregate decline of R371 billion when compared to 30 June 2014." This was largely contributed to local cost pressures, labour action, and continuing downswing in commodity prices. Although, most companies are improving their productivity to address the demanding global and local environment. This is an indication of a highly volatile market space and seems appropriate for the pricing of Asian options. Levy [43] stated that "In markets where prices are prone to extreme volatility the averaging performs a smoothing operation". More recent events, which took place in 2017, indicated no relief to the mining sector. The downgrade of South Africa's sovereign credit rating by Moody's, Fitch, and S & P, and the mining charter of June 2017 escalated uncertainty in the South African economic environment.

Figure 1.1: Plot of the Asian option prices

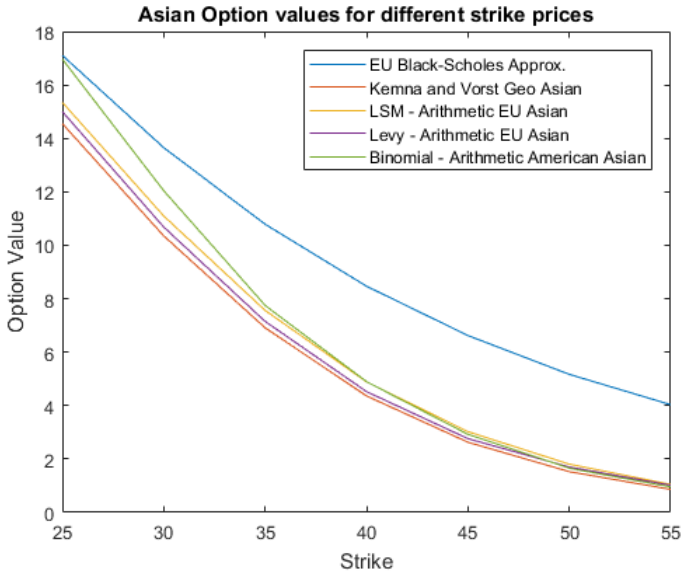
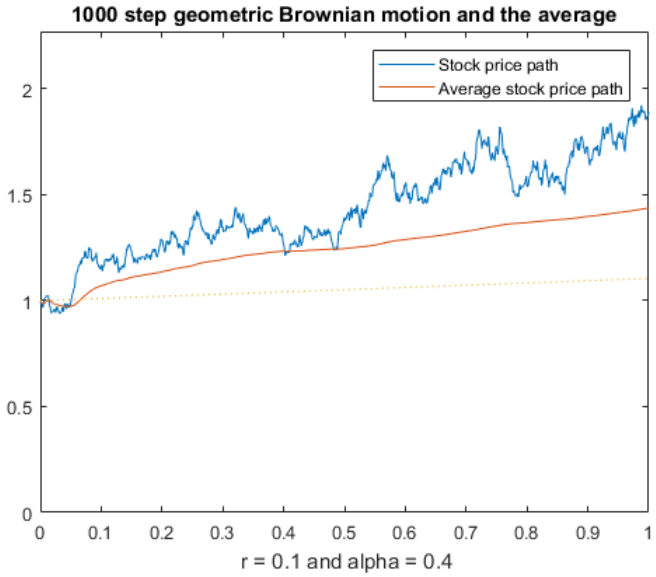


Figure 1.2: Plot of a stock price path and its average



1.1 Option Terminology

Options are defined as two distinct types, namely, put options and call options. A call option is an option that gives the investor the right, but not the obligation, to buy an underlying asset (i.e., stock, commodity, etc.) at a predetermined date and predetermined price. A put option gives the investor the right, but not the obligation, to sell the underlying asset (i.e., stock, commodity, futures, index, etc.) at a predetermined date and predetermined price. This means that the investor or holder of the option has the choice to exercise the option or not.

An option can be bought at a price, which is called a premium. The predetermined price of the contract is called the strike price or exercise price. The date that the option can be exercised or expires is called the maturity, exercise, or expiry date.

Both put and call options have two variants, these are called either European or American options. A European option can only be exercised at maturity, whereas an American option can be exercised anytime before and including maturity. Hence, options have the following payoffs:

$$Call\ option = max[0, S_t - X]\ where\ 0 \leq t \leq T$$

$$Put\ option = max[0, X - S_t]\ where\ 0 \leq t \leq T$$

where T denotes the time at expiry and X denotes the strike price. This means that either at exercise, the option will have a value called the exercise price or will expire worthless.

1.2 Advantages of Asian options

Asian options have the following four main advantages:

1. Asian options are considerably cheaper than standard European and American options with similar maturity and with an averaging period greater than one. This is because an Asian option's volatility of the average price is less than the volatility of the spot price [9].

2. Asian options are exposed to considerably less volatility risk. The averaging of the spot price over time time will lead to lower volatility than the spot at maturity.

3. Asian options are frequently traded on thinly traded assets. Hence, thinly traded asset prices are frequently subject to price manipulation and Asian options offer protection against price manipulation of the underlying, according to Kemna and Vorst [40].

4. In foreign exchange markets, [43] Asian options are offered as Over The Counter derivatives as a means to hedge a stream of foreign currency flows against adverse currency movements. The alternative strategy would be to enter into a strip of individual options, which will be very costly and all that is required to be hedged against the average of the exchange rates.

1.3 Stochastic processes and Arbitrage free pricing

Definition 1 (Forward price of a stock [38])**.** The forward price of a stock is the current stock price denoted by S_0, with an additional return measure, which will attain the cost of holding the stock until time t. Thus the cost of

holding the stock for a period t, is the risk-free interest lost, so the forward price is given by

$$S_0 e^{rt}$$

where r denotes the risk free interest rate.

Definition 2 (Risk Neutral [38])**.** A space is risk neutral if for non-identical assets S_t and for some time period t, the value of the option at $time = 0$ will be the the expected value of the option at $time = t$, discounted with the risk-free interest rate r over the period t. Thus, denoting

$$C(S_t, 0) = e^{-rt}\mathbf{E}^Q[C(P, t)] \qquad \text{(eq:1.3.1)}$$

or

$$C(S_t, 0) = e^{-rt}\mathbf{E}^Q[(S_T - X)^+] \qquad \text{(eq:1.3.2)}$$

where r denotes the compounded risk-free interest rate and P represents the underling asset. Q is the equivalent martingale measure for the discounted underlying and $\mathbf{E}^Q$ is the expectation under this measure.

Definition 3 (Wiener process)**.** Let $(\Omega, \mathcal{F}, \mathbf{P})$ denote a probability space and that all variables are considered to be $\mathcal{F}$ measurable. Then a continuous process X is said to be a Brownian motion or Wiener procress if,

1. $P(X = 0) = 1$

2. For $0 \leq s < t < \infty$, the distribution of $X_t - X_s$ is normally distributed with mean 0 and variance $t - s$

3. For $m \geq 1$, $0 \leq t_0 < t_1 < ... < t_m$ and $Y_1, Y_2, ..., Y_m$ are independent random variables, where $Y = X_{t_i} - X_{t_{i-1}}$

Theorem 1 (Generalized Wiener Process [37])**.** *A generalized Wiener pro-*

cess for a variable x can be defined in terms of dW_t as'

$$dx = adt + bdW_t, \tag{eq:1.3.3}$$

where a and b are constants. adt implies that x has an expected drift rate of a per unit of time, which is regarded as the deterministic component. The bdW_t term can be regarded as adding noise to the path that is followed by x, which is regarded as the stochastic component of the process.

Theorem 2 (The process for a stock price [37])**.** *Following from theorem 1, if we assume that dz is zero, which implies that the process has no uncertainty, the model implies that*

$$\Delta S = \mu S \Delta t. \tag{eq:1.3.4}$$

In the limit $\Delta t \to 0$,

$$dS = \mu S dt. \tag{eq:1.3.5}$$

Integrating between time 0 and T, then yields

$$S_T = S_0 e^{\mu T}. \tag{eq:1.3.6}$$

We then investigate the following process,

$$dS = \mu S dt + \sigma S dz, \tag{eq:1.3.7}$$

or

$$d(\ln S) = \mu dt + \sigma dz. \tag{eq:1.3.8}$$

Equation 1.3.7 is one of the most used stock price processes to model stock price behavior. μ denotes the underlying's rate of return and the variable σ denotes the volatility of the underlying. In a risk neutral world, μ will be replaced by the risk-free rate r.

Chapter 2

Aim of this dissertation

A distinction can be made between the arithmetic and the geometric mean calculated Asian options. We then look at why it would be more beneficial to price Asian options as compared to standard European options. The aim of this dissertation is to effectively price American Asian options, as specified in the Longstaff and Schwartz article. Asian options are inexpensive compared to their European counterparts and Asian options are less prone to volitile price movements. The protection that Asian options offer make them attractive to be structured into hedging policies, where an investor would be exposed to a stream of averaged cash flows. When considering the effective pricing of these Asian options, we will be investigating the Longstaff and Schwartz [39] model to price these options. The Monte Carlo simulation depends on the simulations of stock price paths to accurate price derivatives. The more simulations that are included in the approximation will lead to a more accurate answer. However, the more simulations will lead to more computational effort to compute these approximations. Hence, we implement variance reduction techniques and Quasi random sequences to reduce the number of simulations that are needed to produce a relatively accurate approximate. These techniques will increase the efficiency with which the least squares algorithm is implemented. We aim to measure the

computational time that is used to implement the least squares algorithm.

Some of the other pricing models, that we will be investigating, will require simulation techniques. Such as Monte Carlo simulation which implements variance reduction techniques and Quasi random sequences. We aim to investigate the effectiveness of randomness by generating random numbers from Quasi random sequences versus computer generated random numbers. Quasi random sequences such as the Halton [41], Sobol [50] and Faure [42] sequences will be investigated as well as their impact on simulation pricing. Variance reduction techniques, which are commonly used in Monte Carlo simulations will also be discussed. Measuring the effectiveness of antithetic variates and control variates. All of these simulation aspects will be brought into consideration, when pricing American and American Asian options. The LSM approach uses basis functions to approximate a continuation value. There are different basis functions that can be use. We will, however, focus on using Laguerre, weighted Laguerre, Monomial and Hermite basis functions. Although the LSM method is robust to the choice of basis functions, we investigate how the abovementioned basis functions will influence the pricing of American options.

Finally, we discuss the optimization of the Longstaff and Schwartz model by GPU computation. The GPU has a unique ability. It has multiple cores that can assist in the speedup of computations, even more cores than modern CPUs. We attempt the GPU optimization in Matlab, which is a GPU enabled platform. We also set up Implied and Local volatility surfaces that enable us to price these exotic options quickly, in the real world.

Table 2.1: Topics covered

Discussion topics	Chapters
Black-Scholes approximation for Asian options	6
Binomial tree method for Asian options	7
Generating random numbers via basic LCG	8.2
Sampling methods for Monte Carlo	8.3
Variance reduction techniques	8.4
Quasi random sequences	8.5
Explanation of Longstaff and Schwartz	9
Results for variance reduction and Quasi sequences	10
Investigating the effect of basis functions in LS model	10
Pricing of exotic Asian option with LS model	11
GPU optimization of LS model	12
Implied and Local volatility surfaces for Asian options	13

In this dissertation we have the following research objectives:

1. We investigate different pricing methods for Asian options.(Binomial, Black-Scholes, and Longstaff and Schwartz).

2. We investigate the affect of different basis functions and different Quasi-random sequences on American options, with Longstaff and Schwartz. As well as, multiple variance reduction techniques.

3. We then apply most efficient basis function and Quasi-random sequence to Longstaff and Schwartz to price American Asian options.

4. We also try to optimise the speed of execution by implementing GPU computation on the Longstaff and Schwartz algorithm.

5. We set up Local volatility surface by implementing Dupire's equation to price American asian options with more ease.

Chapter 3

Literature review

The Asian options are fully path dependent. There have been extensive studies the past two decades on both the estimation of geometric Asian options and artithmetic Asian options, since no general analytical solution exists for arithmetic Asian options [40].
In 1990, Kemna and Vorst [40] made a considerable contribution towards the pricing of Asian options. Kemna and Vorst proved that there is no analytical expression for the value of an artithmetic calculated Asian option and deduced that an Asian option will always be equal to, or less than, a standard European option. Kemna and Vorst [40] also provided an analytical expression for a geometrically calculated average, which enabled them to find expressions in the final time interval and over the period from issue date to maturity. The first lattice based model, which was developed for pricing Asian options, was first proposed by Hull and White [2] in 1993. The binomial tree method creates a new type of problem when first implemented to price Asian options. The binomial tree describes the underlying assets price evolution and the possible arithmetic averages grows increases tremendously at lower nodes of the tree. Each of these possible arithmetic averages has to be tracked down and calculated. This means that the number of arithmetic averages grows exponentially when the number of time steps

used to compute this option increases. This makes problems with more time steps increasingly unmanageable. To solve the problem, Hull and White [2] proposed that one should use a set of representative averages at each node within the tree and then employ linear interpolation in an attempt to find the missing values of the option prices. Additional research soon followed by Klassen [11], Barraquand and Pudet [13], and Chalasani [12]. This research proposed a similar pricing technique as Hull and White, with the exception that the set of representative averages will be chosen in a different manner. Lattice models, however, stay one of the most popular models when pricing Asian options because of its efficacy, simplicity, and flexibility. Lattice based models can easily be implemented to price both European and American style Asian options.

Carverhill and Clewlow(1990) [44] numerically evaluated the convolutions of the density functions using the Fast Fourier transform method. The method was very accurate in pricing the option, but computationally inefficient, especially when working with relatively large averages.

Turnbull and Wakeman [45] derived a closed form approximation for a geometrically calculated Asian option as well as an algorithm for the arithmetically calculated Asian option. It claims that the speed of the algorithm can likely be compared to the speed of the Black-Scholes algorithm. Turnbull and Wakeman have found that if the averaging period is smaller than the maturity of the option, and assuming the rate of return is fairly constant. Then the price estimates were similar for the Arithmetic and Geometric Asian options.

According to Levy(1992) [43], Wakeman and Turnbull (1991) recognised the suitability of the log normal distribution as a first order approximation, but overlooked [43] "the fact that when only the first two moments are taken into account in the approximation, the accuracy of the log normal assumption is acceptable making redundant the need to include additional terms in the expansion involving higher moments."

Levy(1992) [43] proposed to approximate the density function of the arith-

metic average. The paper used a straightforward approach, named the Wilkinson approach to approximate the density function of the arithmetic average. This approach has one main advantage, which delivers a closed form analytical expression and is both accurate and easily implemented.

Both the abovementioned, Turnbull and Wakeman(1991) and Levy(1992), models are straightforward, but the models will yield inaccurate approximations under certain assumptions and parameters choices.

Milevsky and Posner(1998) [48] proposed the use of elementary techniques to derive the probability density function of the infinite sum of correlated log-normal random variables and show that the distribution can be fitted to a reciprocal gamma distribution, under some parameter restrictions. "A random variable is reciprocal gamma distributed if its inverse is gamma distributed." They used the reciprocal gamma distribution result to approximate the finite sum of the correlated log-normal random variables and then calculated the arithmetic mean Asian option using the reciprocal gamma distributions as the state price density function. So, Milevsky and Posner proposed a closed form analytical solution for an arithmetic mean Asian option. Thus, when Black and Scholes proposed the use of $N(d_i)$ as their pricing function, which represents the value of the cumulative normal distribution, Milevsky and Posner used $G(d_i)$ in the exact same sense for the cumulative density function, for the gamma distribution.

When using Monte Carlo simulation based models, the backbone of the simulations uses the Black-Scholes (1973) [17] algorithm to the proposed partial differential equation. Furthermore, in the generation of the Monte Carlo simulations, the Boyle, Broadie, and Glasserman (1997) paper states the necessary procedure of implementation, as well as the variance reduction techniques. In this paper, the Quasi random number sequences are discussed. It seems that implementing the Sobol sequences yield the smallest relative estimation error. The following sequences is discussed and tested, Faure(1982) [42], Halton(1960) [41], and Sobol(1967) [50].

Longstaff and Schwartz [39] proposed using the least Squares Monte Carlo (LSM) method for pricing American Asian options. The least squares Monte Carlo method uses backward induction [49] in which a value is recursively assigned to each state at each time step. Then this value is defined as the least squares regression against market price of the option at that specific state and time step. The option value for this regression is defined as the value of exercise possibilities and identifying the value of the time step where exercise would result in. We are now able to evaluate all states at different time steps. The value of the option is calculated by making an optimal decision on option exercise at every time step at hand. This approach is accurate, computationally efficient, and intuitive; it provides the opportunity to price American Asian options. To improve computational efficiency of the LSM model, one can implement a Quasi Monte Carlo method in conjunction with the LSM algorithm.

There is another class of models that solves the governing partial differential equation for Asian options with numerical methods, where both the numerical explicit and implicit finite difference schemes can possibly lead to incorrect Asian option approximations under the evaluation process. It is found that the explicit finite difference scheme is numerically unstable when applied to partial differential equations that model path dependent option pricing. Otherwise, the implicit finite difference scheme is stable, but produces incorrect approximations under certain volatility structures. Wilmott (1993) did a comprehensive study on Asian options, which are approximated by finite difference methods.

Table 3.1: Time progression of pricing Asian options.

1987	Asian options first traded [3].
1990	Kemna an Vorst.
1991	Wakeman and Turnbull.
1992	Levy.
1993	Hull and White - Lattice based method.
1998	Milevsky and Posner.
2001	Longstaff and Schwartz.
2012	Wilkmund.

Chapter 4

Black-Scholes-Merton model introduction

A breakthrough in European stock pricing methods was obatained by Black, Scholes and Merton in the early 1970's [37]. This specific breakthrough was the well renowned development of the Black-Scholes-Merton model or better known as the Black-Scholes model. This model had a significant influence on how to price and hedge derivatives.

4.1 Black-Scholes option pricing formula

Theorem 3. *If we consider a stock which pays no dividends, We follow the proof as propopsed by Turner [38]. The price of a European call option* $C(S_t, X)$ *is given by*

$$C(S_t, X) = S_t\mathcal{N}(d_1) - X\mathcal{N}(d_2)$$

where $\mathcal{N}(.)$ *is the cumulative normal distribution function and* $d_1 = \frac{\ln \frac{X}{S_0} - \mu - \nu^2 T}{\nu\sqrt{T}}$

Proof:

The payoff of a European call is $C(S_T, X) = max(S_t - X, 0)$. By assuming risk neutrality, the expected discounted payoff is $C(S_T, X) = e^{-rT} max(S_t - X, 0)$. We have to remember that $\ln \frac{S_t}{S_0}$ is normally distributed with mean μ and varianve σ, and let the mean of the log normal distribution be located at the forward price of the stock. Then $\sigma = \nu^2 t$ and $\mu = (r - \frac{\nu^2}{2})t$. Note x is a variable and X is the strike price in the following derivation.

$$\begin{aligned}
C(S_0, X) &= e^{-rT}\mathbf{E}[C(S_T, X)] \\
&= e^{-rT}\mathbf{E}[(S_T - X)^+] \\
&= e^{-rT}\int_X^\infty \frac{1}{\sqrt{2\pi T}\sigma x}(x - X)e^{\frac{-(\ln \frac{x}{S_0} - \mu)^2}{2\sigma^2 T}} dS \\
&= e^{-rT}\int_X^\infty \frac{1}{\sqrt{2\pi T}\sigma x} x e^{\frac{-(\ln \frac{x}{S_0} - \mu)^2}{2\sigma^2 T}} dx - e^{-rT}\int_X^\infty \frac{1}{\sqrt{2\pi T}\sigma x} X e^{\frac{-(\ln \frac{x}{S_0} - \mu)^2}{2\sigma^2 T}} dx \\
&= e^{-rT}\int_X^\infty \frac{1}{\sqrt{2\pi T}\sigma} e^{\frac{-(\ln \frac{x}{S_0} - \mu)^2}{2\sigma^2 T}} dx - e^{-rT}\int_X^\infty \frac{1}{\sqrt{2\pi T}\sigma x} X e^{\frac{-(\ln \frac{x}{S_0} - \mu)^2}{2\sigma^2 T}} dx
\end{aligned} \tag{eq:4.1.1}$$

Looking at the first part of the integral and now setting $z = \frac{\ln \frac{x}{S_0} - \mu}{\nu\sqrt{T}}$ then $dz = \frac{dx}{x\nu\sqrt{T}}$. The first part simplifies as follows,

$$e^{-rT} S_0 e^{\mu + \frac{\nu^2 T}{2}} \int_A^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} dx \tag{eq:4.1.2}$$

with $A = \frac{\ln \frac{X}{S_0} - \mu - \nu^2 T}{\nu\sqrt{T}}$

Hence, by recognizing that the integral represents the cumulative distribution function for the standard normal variable, we have that,

$$\begin{aligned}
S_0(1 - \mathcal{N}(\frac{\ln \frac{X}{S_0} - rT - \frac{\nu^2 T}{2}}{\nu\sqrt{T}}) &= S_0\mathcal{N}(-\frac{\ln \frac{X}{S_0} - rT - \frac{\nu^2 T}{2}}{\nu\sqrt{T}}) \\
&= S_0\mathcal{N}(\frac{\ln \frac{X}{S_0} + rT + \frac{\nu^2 T}{2}}{\nu\sqrt{T}})
\end{aligned}$$

Thus, this yields the first term for the proof.
To complete the proof we now examine the second part of the pricing equation: Let $z = \frac{\ln \frac{x}{S_0} - \mu}{\nu\sqrt{T}}$ then $dz = \frac{dx}{x\nu\sqrt{T}}$ and,

$$\begin{aligned} -e^{-rT}\int_X^\infty \frac{1}{\sqrt{2\pi}\nu x} X e^{\frac{(\ln \frac{x}{S_0} - \mu)^2}{2\nu^2 T}} &= -e^{-rT}\int_{A+\nu\sqrt{T}}^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} dz \\ &= -e^{-rT} X(1 - \mathcal{N}(A + \nu\sqrt{T})) \\ &= -Xe^{-rT}(\mathcal{N}(-A - \nu\sqrt{T})) \\ &= -Xe^{-rT}(\frac{\ln \frac{X}{S_0} + rT - \frac{\nu^2 T}{2}}{\nu\sqrt{T}}) \end{aligned}$$

which yields the second term of the equation and therefore completes the proof.

4.2 Black-Scholes model on European option pricing

In 1973, a paper was published by Fischer Black and Myron Scholes and their derivative pricing theory was based on a Geometric Brownian process. Now from Theorem 2 in the previous section, we say that at some time t and underlying asset price S_t comprise a Geometric Brownian motion if it satisfies the following stochastic differential equation.

$$dS_t = \mu S_t dt + \sigma S_t dW_t \qquad \text{(eq:4.2.1)}$$

Where the W_t represents the Wiener process or Brownian motion, μ is the percentage drift and σ represents the volatility. The $\mu S_t dt$ term is known as the deterministic term and $\sigma S_t dW_t$ is known as the stochastic term that generates the noise of the process.

With the use of Itô's Lemma we are able to find an analytical expression for

the abovementioned stochastic differential equation. That is,

$$S_t = S_0 e^{((\mu - \frac{\sigma^2}{2})t + \sigma\sqrt{t}Z)}$$

for an arbitrary initial stock value S_0 and where Z represents the standard normal distribution. From this expression we can see that $log(\frac{S_t}{S_0})$ is normally distributed with mean $(r - \frac{\sigma^2}{2})t$ and variance $\sigma^2 t$. Hence,

$$S_t = S_0 e^{((r - \frac{\sigma^2}{2})(t) + \sigma\sqrt{t}Z)}.$$

The Black-Scholes pricing formula, as stated and proved above, gives the exact value of a European call or put option. This is where we see that American options have no closed form solution. The price of a European call will be denoted by C and a European put will be denoted as P on a nondividend paying underlying asset. We assume that the initial value is denoted by S_0 and can therefore be calculated by the following formulas:

$$C = S_0\mathcal{N}(d_1) - Ke^{-rT}\mathcal{N}(d2),$$

and

$$P = Ke^{-rT}\mathcal{N}(-d2) - S_0\mathcal{N}(-d_1,)$$

where

$$d_1 = \frac{\ln \frac{S_0}{K} + (r - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}},$$

and

$$d_2 = d1 - \sigma\sqrt{T}.$$

and $\mathcal{N}$ denotes the cumulative distribution function of the standard normal distribution, $N(0, 1)$.

4.3 Assumptions of the Black-Scholes model

Assumptions for the Black-Scholes formula are as follows, stated by Hull [37], and Winarti, Guna, and Noviyanti [6]:

1. The stock price process is stated with a constant μ and σ.

2. The short selling of securities with full use of the proceeds is permitted.

3. There are no transactional costs or taxes. All securities are perfectly divisible.

4. There are no dividends during the life of the derivative.

5. There are no riskless arbitrage opportunities.

6. Security trading is continuous.

7. The risk-free rate of interest, r, is constant and the same for all maturities.

For our purposes we assume that σ and r are deterministic by nature. However, it is also possible that σ and r are known functions of t. Interest rates can be stochastic of nature, but under the assumption that the stock price distribution at maturity of the option is still log normal [37].

4.4 Black-Scholes for Asian Options

As we discussed in theorem 3, we are able to find a closed form formula for pricing a standard European option, on a nondividend paying stock. We now follow Wiklund [21] [2012] and observe a formula for estimating both arithmetic and geometric Asian options. Asian option pricing, in terms of the Black-Scholes framework, has already been researched and developed by

Kemna And Vorst [40], Levy [43], Turnbull and Wakeman [45], Milevsky and Posner [27], and Curran [20], to name a few. We investigate the pricing of geometric Asian options in section 6.1 and discuss the pricing of arithmetic Asian options in 6.2.

4.5 Geometric Asian option approximation formula

As we discussed earlier we are able to find a closed form approximation formula for the geometric Asian option. This is only possible because the stock price process and the geometric average have a critical property in common. They both have log normal distributional properties, which we showed earlier in this dissertation. We follow Wiklund [21], and Winarti, Guna and Noviyanti [6]. The log normal for the geometric mean will not be proven here as it does not form an integral part of this disserataion. But the pricing of geometric Asian options are thoroughly discussed in the papers of Wiklund [21], and Winarti, Guna and Noviyanti [6]. We will show the pricing formula for a non-dividend paying stock as,

$$\begin{aligned} C(S_0, K, A_j, T) &= S_0 A_j N(d_{n-j} + \sigma\sqrt{T_{2,n-j}}) - Ke^{-rT} N(d_{n-j}) \\ P(S_0, K, A_j, T) &= Ke^{-rT} N(-d_{n-j}) - S_0 A_j N(-d_{n-j} - \sigma\sqrt{T_{2,n-j}}) \end{aligned} \quad \text{(eq:4.5.1)}$$

where n denotes the number of observations that form the average, j represents the number of observations that was passed in the averaging period and h represents the observation frequency. C is the price of a call option and P is the price of a put option and $N(u)$ is the cumulative probability distribution function for a standardized normal distribution. S_0 represents the stock price at time 0, K represents the strike price, r is the risk-free rate of return, σ is the underlying stocks volatility and T represents the time to

maturity of the option.
where

$$
\begin{aligned}
d_{n-j} &= \frac{ln(\frac{S_0}{K}) + \big(r - \frac{\sigma^2}{2}\big)T_{1,n-j} + ln(B_j)}{\sigma\sqrt{T_{2,n-j}}} \\
A_j &= e^{\frac{-r(T-T_{1,n-j})-\sigma^2(T_{1,n-j}-T_{2,n-j})}{2}} B_j \\
T_{1,n-j} &= \frac{n-j}{n}(T - \frac{(n-j-1)h}{2}) \\
T_{2,n-j} &= \big(\frac{n-j}{n}\big)^2 T - \frac{(n-j)(n-j-1)(4n-4j+1)}{6n^2}h \\
B_j &= \big(\prod_{j=1}^{n} \frac{S(T-(n-j)h)}{S}\big)^{\frac{1}{n}} \\
B_0 &= 1
\end{aligned}
\tag{eq:4.5.2}
$$

Note, if we have that $n = 1$ and $j = 0$, this formula will just reduce to the normal Black-Scholes pricing formula. For further results and explanation of the derivation, refer to Wiklund [21], and Winarti, Guna and Noviyanti [6].

4.6 Arithmetic Asian option approximation formula

We have that the arithmetic mean does not follow a log normal distribution and does not satisfy the normality assumption of the Black-Scholes framework. For this reason we are unable to obtain a closed form formula to price arithmetic Asian options. But we can ultimately work around this shortcoming by approximating the arithmetic mean by using the geometric mean. Then it is possible to approximate an estimate for an arithmetic Asian option. If we want to find an estimate for an arithmetic Asian option, we can approximate the value of this option by conditioning it to the geometric mean price of the underlying asset. We once again follow the papers of

Wiklund [21], and Winarti, Guna and Noviyanti [6] for the proof of the arithmetic Asian option approximation. Hence,

$$C(S_0, K, A, T) = e^{-rT} E[(A - K)^+]$$

and

$$P(S_0, K, A, T) = e^{-rT} E[(K - A)^+]$$

where A is the arithmetic mean [20] and G is the geometric mean of the respective underlying asset. We follow the proof and reasoning of Winarti, Guna and Noviyanti [6] for the next section. If we now consider the case where the averaging has not yet started, the the price of an arithmetic Asian call option can be expressed as,

$$\begin{aligned} C &= e^{rT} E[(A - K)^+] \\ &= e^{rT} E[(A - K)^+ | G] \\ &= e^{rT} \int_0^\infty E[(A - K)^+ | G = x] g(x) dx \end{aligned} \tag{eq:4.6.1}$$

where g is denoted as the log normal density function of G. We now define our price C in terms of C_1 and C_2 as in and out of the money approximations. So,

$$C = e^{-rT}(C_1 + C_2)$$

, where

$$C_1 = \int_0^K E[max(A - K, 0) | G = x] g(x) dx$$

and

$$C_2 = \int_K^\infty E[max(A - K, 0) | G = x] g(x) dx$$

.

Since the arithmetic average is greater than the geometric average ($A \geq G$),

so:

$$C_2 = \int_K^\infty E[(A-K)|G=x]g(x)dx$$

.

However, C_1 is the situation where the geometric option is out of the money and the respective arithmetic option is unknown. C_2 is the situation, where both options are out of the money. The C_1 approximation is then as follows:

$$C_1 = \int_L^K E[A-K|G=x]g(x)dx$$

with $L = \{x|E(A|G=x) = K\}$ Then we have

$$\begin{aligned} C &= e^{-rT}(C_1 + C_2) \\ C_1 + C_2 &= \int_L^\infty E[A-K|G=x]g(x)dx] \\ &= \frac{1}{n}\sum_{i=1}^n \int_L^\infty E[S_{t_i}|ln(G) = ln(x)]g(x)dx - K\int_L^\infty g(x)dx \end{aligned} \tag{eq:4.6.2}$$

with $K\int_L^\infty g(x)dx = KN(\frac{\mu_G - ln(L)}{\sigma_G})$
The approximation for an arithmetic Asian call option can be expressed as;

$$\begin{aligned} C(S_0,K,A,T) &\approx e^{-rT}\left[(\frac{1}{n}\sum_{i=1}^n e^{\frac{\mu_i+\sigma_i^2}{2}}N(\frac{\mu_G - ln(L)+\gamma_i}{\sigma_G})) - KN(\frac{\mu_G - ln(L)}{\sigma_G})\right], \\ P(S_0,K,A,T) &\approx e^{-rT}\left[KN(-\frac{\mu_G - ln(L)}{\sigma_G}) - (\frac{1}{n}\sum_{i=1}^n e^{\frac{\mu_i+\sigma_i^2}{2}}N(\frac{\mu_G - ln(L)+\gamma_i}{\sigma_G}))\right] \end{aligned} \tag{eq:4.6.3}$$

where furthermore

$$\begin{aligned}
\mu_G &= ln(S_0) + (r - \frac{\sigma^2}{2})(t_1 + (n-1)\Delta t/2) \\
\mu_i &= ln(S_0) + (r - \frac{\sigma^2}{2})(t_1 + (i-1)\Delta t) \\
\sigma_i &= \sigma\sqrt{(t_i)} \\
\sigma_G^2 &= \sigma^2 h \frac{(2n+1)(n+1)}{6n} \\
L &= \{x | E(A|G=x) = K\} \\
\gamma_i &= \frac{\sigma^2 h}{2n}((2n+1)i - i^2)
\end{aligned} \tag{eq:4.6.4}$$

where t_1 is the time to the first averaging point, Δt represent the time between averaging points and other parameters are the same as defined before.

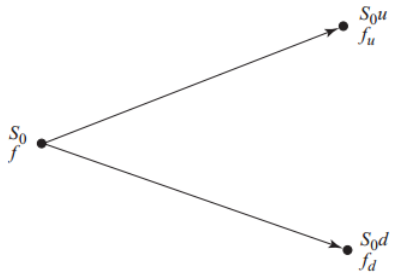
Chapter 5

Tree based Method for American Asian options

5.1 Brief introduction into Binomial model

The following will be a brief introduction into binomial stock trees. The next figure[1] 5.1 shows an elementary one step tree. We assume a no arbitrage argument.

Figure 5.1: Elementary representation of a one step tree model



1

[1]John C Hall, 2012. Options, Futures, and Other Derivatives (Global Edition). 8th Edition. Pearson. Page 255, Figure 12.2

S_0 denotes the stock price at time zero and f is the current price of an option that derives its value from the underlying stock. An option can be based on an underlying that is tradable or not. The total length of time will be denoted by T and each time step will be denoted by Δt. The stock price can move up or down from the current value of the stock S_0. So, the stock price can move to the "upstate" or "S_0u" with a factor u or move to a "downstate" or "S_0d" with a factor d. Just note that it logically makes sense that $u > 1$ and $d < 1$. The value of the stock in the "upstate" is $S_0 u$ and so for the downstate is $S_0 d$. The corresponding value for the option in the upstate and downstate is f_u and f_d.

Firstly, we look at some derivations without the risk of default:
From figure 5.1 we have that: $pS_0u + (1-p)\,S_0d = e^{(r-q)\Delta t}S_0$ by equating the expected value of the share price at time Δt or one time step, to the value of the share price at time zero denoted by S_0 times the risk-free rate minus the coupon rate.
Note that p is the probability to move to the upstate and $(1-p)$ is the probability to move to the downstate. p is a subjective probability which will differ from investor to investor. r denotes the riskless expected return and q is the yield rate of the asset that is paid annually or better known as a dividend rate. The capital gain will provide a return of $(r-q)$. If we solve for p we get:

$$\begin{aligned} pS_0u + \ S_0d - pS_0d &= e^{(r-q)\Delta t}S_0 \\ pS_0\,(u-d) &= S_0(e^{(r-q)\Delta t} - d) \\ \therefore p = \frac{(e^{(r-q)\Delta t} - d)}{(u-d)} &= \ \frac{l-d}{u-d} \end{aligned} \qquad \text{(eq:5.1.1)}$$

Where $l = e^{(r-q)\Delta t}$.
Substituting p into $E\,[S_T] = \ pS_0u + (1-p)\,S_0d$ we obtain

$$
\begin{aligned}
E\left[S_T\right] = &\ pS_0u + S_0d - pS_0d \\
&= pS_0(u-d) + S_0d \\
&= \frac{(e^{(r-q)\Delta t} - d)}{(u-d)} S_0(u-d) + S_0d \qquad \text{(eq:5.1.2)} \\
&= (e^{(r-q)\Delta t} - d)S_0 + S_0d \\
Hence,\ E\left[S_T\right] &= e^{(r-q)\Delta t}
\end{aligned}
$$

If the asset yield rate is disregarded for a moment then $E\left[S_T\right] = e^{r\Delta t}$. This means that for valuation purposes we assume the following:[2]

1. It is assumed that the expected return from all assets is the risk-free rate.

2. Discounting occurs at the risk-free rate for valuing the derivative pay-offs.

The volatility σ of a stock price is defined so that $\sigma\sqrt{\Delta t}$ is the standard deviation of the return on the stock price with length of each time step Δt. Then the variance of the stock return is:

$$
pu^2 + (1-p)\,d^2 - \left[pu + (1-p)\,d\right]^2 = \ \sigma^2\Delta t
$$

Substituting p into the above mentioned variance. We then obtain:

$$
\frac{(e^{(r-q)\Delta t} - d)}{(u-d)} u^2 + \left(1 - \frac{(e^{(r-q)\Delta t} - d)}{(u-d)}\right) d^2
$$

$$
- \left[\frac{\left(e^{(r-q)\Delta t} - d\right)}{(u-d)} u + \left(1 - \frac{\left(e^{(r-q)\Delta t} - d\right)}{(u-d)}\right) d\right]^2 = \sigma^2\Delta t
$$

[2]John C Hall, 2012. Options, Futures, and Other Derivatives (Global Edition). 8th Edition. Pearson. Page 428, Risk neutral valuation

Now using $Var\,(x) = E\,[x^2] - E[x]^2$ and by substituting p and then solving for u and d. The result is found with the Taylor expansion that a solution for u and d are:

$$u = e^{\sqrt{\sigma^2 \Delta t}}$$

and

$$d = e^{-\sqrt{\sigma^2 \Delta t}}$$

or

$$d = \frac{1}{u}$$

These are the values proposed by Cox, Ross and Rubenstein (1979) for matching volatility. Assumptions made when using the binomial tree:

1. The binomial tree is recombining
2. Constant volatility

5.2 Pricing American Asian options with Tree based models

In 1993, Hull and White [2] proposed a binomial tree based pricing method for Asian options. This section will briefly discuss this pricing process, but will not be proven. The binomial tree based model is able to track all possible arithmetic average prices at each specific node in the tree. Hence, this tree based method is able to derive exact option prices for both arithmetic and geometric Asian options. We closely follow the model as specified by M. Costabile [15] We first consider a European call option, which is written on the arithmetic average of the prices of the underlying with $S(t) = S$ at any specified time t. At maturity T, the European option will have the payoff of $max(A(T) - K, 0)$. Where the $A(T)$ denotes the average of the underlying

asset prices during the lifetime of the option and K denotes the strike price. This European option is known as a fixed strike Asian option, as we have discussed in previous sections of this dissertation. Now we are able to define the arithmetic average of the underlying asset prices, which are attained during the lifetime of the option as a minimum and maximum average, for j consecutive down movements and $i-j$ consecutive up movements. So, we have

$$\begin{aligned} A_{max}(i,j) &= \frac{S(1+u+u^2+...+u^{i-j}+u^{i-j}d+u^{i-j}d^2+...+u^{i-j}d^j)}{i+1} \\ &= \frac{(S\frac{1-u^{i-j+1}}{1-u}+Su^{i-j}d\frac{1-d^j}{1-d})}{i+1} \end{aligned} \tag{eq:5.2.1}$$

and

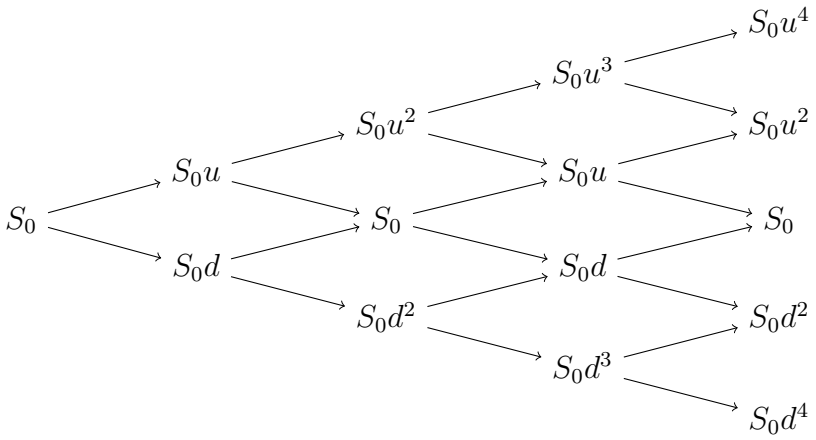
$$\begin{aligned} A_{min}(i,j) &= \frac{S(1+d+d^2+...+d^j+d^ju+d^ju^2+...+d^ju^{i-j})}{i+1} \\ &= \frac{(S\frac{1-d^{j+1}}{1-d}+Sd^ju\frac{1-u^{i-j}}{1-u})}{i+1} \end{aligned} \tag{eq:5.2.2}$$

Then for each node (i,j) the average prices are captured in an array and arranged from maximum to minimum arithmetic averages. The arithmetic averages for each node (i,j) are then calculated as

$$A(i,j,k) = \frac{M-k}{M}A_{max}(i,j) + \frac{k}{M}A_{min}(i,j) \;\; for \;\; k = 0,1,2,...,M$$

where M represents the number of averages at any specific node.

Figure 5.2: 4 time step binomial tree model



For simplification of the process and illustration purposes let us consider the four time step binomial model as seen above. By considering the example illustrated by Dyakopu [16], we will denote the nodes as $N(t, j)$, where t indicates the time step and j indicates the tranches from high to low. We now consider one of the terminal nodes, say N(4,4). There is, however, only one possible trajectory to calculate the average for $N(4, 4)$ and that is the path $(S_0, S_0u, S_0u^2, S_0u^3, S_0u^4)$. But if we consider say another node, for example at $N(4, 2)$, then the first and maximum average can be calculated on the path $(S_0, S_0u, S_0u^2, S_0u, S_0)$ and the average set is $A(4, 2; 1) = A_{max}(4, 2)$. We also note that $Su^2 = S_{max}(4, 2; 1)$ is the highest recorded value of the path. The following four averages are calculated from the next paths: $(S_0, S_0u, S_0, S_0u, S_0)$, $(S_0, S_0d, S_0, S_0u, S_0)$, $(S_0, S_0d, S_0, S_0d, S_0)$ and $(S_0, S_0d, S_0d^2, S_0d, S_0)$. Hence, N(4,2) has a set of representative averages, which are all true averages, except the one average generated by the path $(S_0, S_0d, S_0, S_0u, S_0)$. By using backward induction we are able to approximate a estimate for the option by considering the

representative averages in such a way that:

$$A_u = \frac{(i+1)A(i,j,k) + S_0 u^{i+1-j} d^j}{i+2}$$

$$\Rightarrow \frac{(i+1)A(i,j,k) + S(i,j)u}{i+2}$$

and

$$A_d = \frac{(i+1)A(i,j,k) + S_0 u^{i+1-(j+1)} d^{j+1}}{i+2}$$

$$\Rightarrow \frac{(i+1)A(i,j,k) + S(i,j)d}{i+2}$$

where A_u is inside the range of $[A(i+1,j,k_u), A(i+1,j,k_u-1)]$ and A_d is in the range of $[A(i+1,j+1,k_d), A(i+1,j+1,k_d-1)]$. These sets represent the set of representative averages at each node. By using linear interplolation we are now able to approximate the corresponding value C_u with

$$C_u = w_u C(i+1,j,k_u) + (1-w_u)C(i+1,j,k_u-1)$$

where

$$w_u = \frac{A(i+1,j,k_u) - A_u}{A(i+1,j,k_u-1) - A(i+1,j,k_u)},$$

$$C_d = w_d C(i+1,j,k_d) + (1-w_d)C(i+1,j,k_d-1),$$

$$w_d = \frac{A(i+1,j,k_d) - A_d}{A(i+1,j,k_d-1) - A(i+1,j,k_d)}.$$

Our option price is then calculated as:

$$C(i,j,k) = [pC_u + (1-p)C_d]e^{-t\Delta t}$$

American options then have the following payoff:

$$C(i,j,k) = max[A(i,j,k) - K; (pC_u + (1-p)C_d)e^{-t\Delta t}].$$

5.3 Example: Arithmetic Asian option

We consider the following example for European and American Asian options:

Table 5.1: Example assumptions

S_0	40
σ	25%
r	6%
Time base	252
K	24-56 by 1 increments
Time to maturity	1 year
Exercised per year	36(weekly)

Figure 5.3: Plot of European and American Asian options

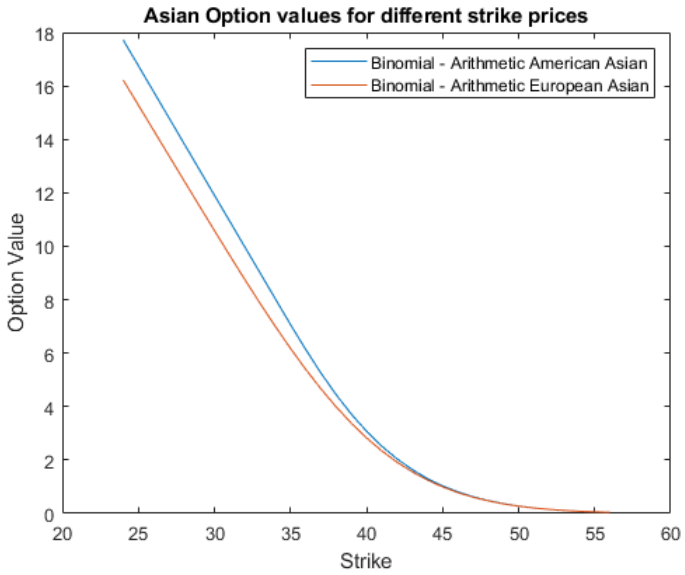


Table 5.2: American and European Asian option values with the Binomial tree method

Strike	American value	European value
24	17.725	16.222
25	16.755	15.280
26	15.785	14.338
27	14.816	13.397
28	13.846	12.457
29	12.877	11.520
30	11.909	10.587
31	10.940	9.664
32	9.972	8.754
33	9.004	7.865
34	8.037	7.003
35	7.070	6.177
36	6.140	5.395
37	5.259	4.665
38	4.448	3.991
39	3.719	3.379
40	3.078	2.830
41	2.521	2.345
42	2.046	1.923
43	1.644	1.560
44	1.309	1.252
45	1.033	0.996
46	0.808	0.783
47	0.626	0.610
48	0.481	0.471
49	0.366	0.360
50	0.277	0.273
51	0.207	0.205
52	0.154	0.152
53	0.113	0.112
54	0.083	0.082
55	0.060	0.060
56	0.043	0.043

Chapter 6

Monte Carlo

For the purposes of this paper we will only be considering the application of Monte Carlo methods in a financial derivative pricing setting. We let the relative option price be denoted by μ and then written as the integral that represents the expectation of the payoff of the financial derivative under the risk neutral probability measure.

The general form of a Monte Carlo integral can be approximated by a discrete function. We can then consider the numerical integration in the dimension s. But for simplicity let us assume $s = 1$ and consider the numerical integral in the one-dimensional space with the trapezoidal rule over the unit interval [0,1]. According to Niederreiter, (1992) we obtain the following result [30],

$$\int_0^1 f(u)du \approx \sum_{n=0}^{m} w_n f(\frac{n}{m})$$

where $m \geq 0$ and w_n is given by $w_0 = w_m = \frac{1}{2m}$ and $w_n = \frac{1}{m}$ for $1 \leq n \leq m-1$.

Now, in a multi-dimensional setting and considering the expectation of the option price. The option price will be denoted by an integral of μ, which represents the discounted payoff in the risk neutral probability setting. So

then we have that according to [31]

$$\mu(f) = \int \cdots \int_0^1 f(u_1, \ldots, u_n)\, du_1 \ldots du_n$$

$$\mu(f) = \int^{[0,1)^n} f(\overline{u}) d\overline{u}$$

$$\mu(f) = \mathbf{E}[f(\mathbf{U})],$$

Where $I_n = [0,1)^n$ is a closed n-dimensional unit cube and $[0,1)^n =$ $u = 0 \leq u_k < 1$; for k=1,2,,n. The function f, has the domain $[0,1)^n$ and range $\mathbb{R}$. Thus, $f : [0,1)^n \to \mathbb{R}$ the function f plots a random point $u_i, for\ i = 1, 2, , n$, which is in the unit cube $I_s = [0,1)^n$ and each of these $u_i's$ has a uniform $[0,1)$ distribution.

Monte Carlo [36] in general, is not a competitive method for calculating one-dimensional integrals, but Monte Carlo obtains a $O(n^{-\frac{1}{2}})$ convergence rate, not just over the unit interval. For the estimation over higher dimensions as outlined above, we also change f and σ_f. The standard error will still have the form $\frac{\sigma_f}{\sqrt{n}}$ for the Monte Carlo estimate based on n draws in the $[0,1]^s$ domain and the $O(n^{-\frac{1}{2}})$ convergence rate for all dimensions s. Thus Monte Carlo methods are far more competitive in higher dimensions.

In this case, where f represents the pricing function of an option, under the risk neutral probability measure. It will be sufficient to specify the relation between the standard normal random variable and the Monte Carlo method that will be explained in detail.

Now, if we have the function k of n independent standard normal random variables. Where k now represents the payoff function of a claim, for example $Z_1, ..., Z_n$, then we are able to get $f(u) = k[\Phi^{-1}(u_1), ..., \Phi^{-1}(u_n)]$, where Φ^{-1} indicates the inverse function of the standard normal random variable. The distribution of the standard normal random variable is specified as

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_t^{-\infty} e^{(-z^2)/2} dz$$

Then the Monte Carlo estimate for the expected value $\mathbf{E}[f(\mathbf{U})]$ is obtained by taking N independent Uniform distributed random samples $u = 0 \leq u_k < 1$; for $k = 1, 2, , n$ and letting

$$\mu(f) = \mathbf{E}[(f\mathbf{U})] \approx \frac{1}{j} \sum_{i=1}^{j} f(U_i)$$

Where j represents the number of replications that are done. For the estimation of complex problems the basic Monte Carlo approach may provide results with large errors, because the estimation error is $O(1/n)$.
Boyle, Broadie and Glasserman (1997) came to the following conclusion regarding the difficulties of Monte Carlo Simulation. "The following points must be viewed as deficiencies of the Monte Carlo method for numerical integration:

(i) There are only probabilistic error bounds;

(ii) The regularity of the integrand is not reflected;

(iii) Generating random samples is difficult.

Practitioners avoid the difficulty of generating random samples by using Pseudo-random numbers or Quasi random numbers instead of "truly" random samples. This is done because random samples are a statistical concept that is only defined as part of a whole ensemble of samples." Thus, it is difficult to define a truly random sample. Therefore, later on we look at variance reduction techniques and Quasi sequences.

6.1 Generating random numbers

At the center of almost all Monte Carlo simulations [36] there is the generation of seemingly random numbers. We treat these sequences as genuinely random sequences in our Monte Carlo estimation. The following

section follows from Monte Carlo methods in Financial Engineering by Paul Glasserman. Due to this assumption of "genuinely random sequences", it enables us to use aspects from probability- and statistical- theory. Pseudo-random sequences are sufficient in generating random sequences to make sense of analyses that are performed. Nonetheless, we always know that these algorithms are deterministic by nature, which delivers seemingly random sequences.

Before we continue with the discussion of random sequences, we discuss what properties a random sequence ideally should comprise. For some random variables $U_1, U_2, ...$ should have the following properties,

i.) $U_k, \forall k \in \mathbb{Z}$ in Uniformly distributes between 0 and 1,

ii.) all $U_k, \forall k \in \mathbb{Z}$ are mutually independent

For property (i), values between 0 and a $\frac{1}{4}$ would be equally beneficial and can easily be normalized to be distributed between 0 and 1. Uniform random variables over the unit interval $[0, 1]$ can be easily transformed into samples from any other distribution. Property (ii) is more essential to random number generation. It implies that generated pairs should be uncorrelated from each other and that any specific U_k should not be predictable from the sequence $U_1, U_2, ..., U_{k-1}$

Hence, a random generator usually generates a finite number of random numbers $u_1, u_2, ..., u_k$ over the unit interval. Input parameters for the generating algorithms are usually specified by the user.

So, we obtain a set $U_1, U_2, ..., U_k$ of independent uniforms. An effective random number generator then produces sequences that are consistent with properties (i) and (ii). If we then generate a large number of values or k, then the fraction of the values that fall into a sub-interval of the unit interval, should be approximated by the length of the sub-interval. This is a good indication of uniformity of the sequence that is generated. To determine independence between the sequence values, there should be no discernible

pattern between the values $u_1, u_2, ..., u_k$.
The basic form of a random number generator is demonstrated by

$$a_{i+1} = f(a_i); u_{i+1} = h(a_{i+1}) \qquad \text{(eq:6.1.1)}$$

for some deterministic functions f and h. a_i represents the sequence on numbers produced by the deterministic function, f.

6.2 Random number generation

6.2.1 Linear Congruential Generators (LCG)

The Linear congruential generator is a recurrence of the following form:

$$a_{i+1} = k\, a_i\, mod(m) \qquad \text{(eq:6.2.1)}$$

$$u_{i+1} = \frac{a_{i+1}}{m} \qquad \text{(eq:6.2.2)}$$

where the multiplier is denoted by k and the modulus m. These values are integer constants for the values that are generated, for some given seed value a_0. The seed integer should be chosen by the user between 1 and $m-1$. So, the modulus operator will return the remainder of $\frac{ka_i}{m}$. For example, 9 mod 4 is 1; 8 mod 3 is 2; 3 mod 5 is 3, etc. [36] "Because the result of the mod m operation is always an integer between 0 and $m-1$,so the output values u_i is produced and will always be between 0 and $\frac{m-1}{m}$ in particular, they lie in the unit interval."

When constructing a random number generator, there are considerations that need to be taken account, such as computational speed, portability, randomness, duplicability, and period length.

* **Computational Speed:** The reason that a random number generator needs to be efficient is that the generator is called thousands or even millions of times, when generating a sequence of numbers. Hence, the Linear Congruential generator algorithm is one of the fastest generators due to its simplicity, compared to other random number generators that we will discuss later in this paper. These "less efficient" random number generators use more computational time per value that is generated for the sequence and makes the generator inherently slower than a Linear Congruential generator. There are, however, techniques to save computational time by carefully selecting your choice of input or starting parameters. In our case, as seen in (eq:6.1.1) and (eq:6.1.2), we can choose m to be of the power 2, which contributes to the efficiency of the number generation by simplifying the mod function to a shift in computation. If we consider a random number generator with poor distribution properties, the infinitesimal speedups gained by carefully choosing the parameters do not significantly contribute towards the speed of the generator.

* **Portability:** An algorithm for random number generators should be able to produce the same sequence of values on different computing platforms. In the design of certain algorithms, the objective of speed and period length sometimes are driven by machine specific properties. Machine specific properties such as thermal noise, photoelectric effect, or some other quantum phenomenon. Usually, these phenomena are unpredictable in theory and recorded by a transducer and converted to an electric signal. Hence, it is unlikely to replicate such a sequence on another machine.

* **Randomness:** The most desired property to obtain from a random number generator is usually the hardest to define and ascertain. There are two main factors present that drive randomness: theoretical properties

and statistical randomness tests. Lots of research has already been conducted of the generation of points of most of the widely known random point generators. These research articles help a lot to narrow down parameter input values for these generators to be certain of efficacy in parameter choices. When these random number generators have been identified with good theoretical properties, then the generated sequence can come under statistical scrutiny. The statistical scrutiny is to be certain that the generated sequence does not deviate from randomness. There is, however, an upside to the field, namely that it has been thoroughly researched and one or more of these random number algorithms can be applied to most applications in various fields.

* **Duplicability:** When one considers the generation of randomness from a physical process that is performed, such as the computer's clock speed or some other related process. One major drawback from incorporating such a process is the inability to reproduce the randomness that was created. In some circumstances it would be beneficial to reproduce the exact same type of randomness, to be able to access the impact of the randomness on calculations that depend on this "created randomness". It is also easier then to compare simulations with different input arguments and access the impact thereof. This type of Duplicability is seen when working with the Linear Congruential (LCG) generator and is reproduced easily using the same seed value, which is denoted by x_0.

* **Period length:** As seen in the following example, any random number generator that has the same form as seen in (eq. 6.2.1) will eventually reach a specific point where the random number generator starts to repeat itself. The best type of random number generator is one that produces many distinct values before repeating itself. For example the Linear Congruential generator will create $n - 1$ distinct values, with

modulus (n) before repeating itself. If a Linear Congruential generator creates a full period, meaning $n-1$ distinct values, then the spacing between the generated values that we obtain u_i, will have a width of $\frac{1}{n}$. We deduce logically as n increases the spacing of these randomly generated values will be closer to each other. With a larger n the better approximation will be a uniform distribution.

Example 6.2.1. Let us consider the following LCG;

$$a_{i+1} = k\, a_i\, mod(m)$$

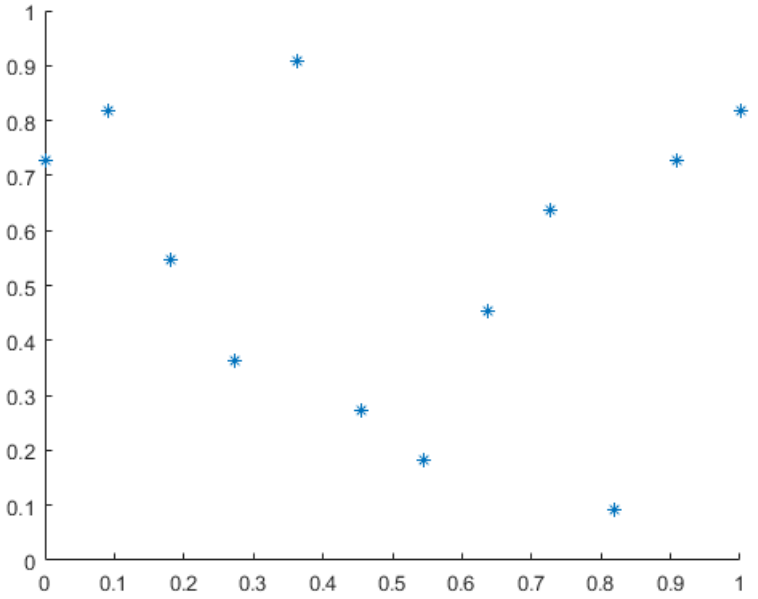
$$u_{i+1} = \frac{a_{i+1}}{m}$$

where:
k = 8
m = 12
$a_0 = 1$
When we apply this basic algorithm recursively, we get to the following sequence;

$$8, 9, 6, 4, 10, 3, 2, 5, 7, 1, 8, 9$$

We observe that once a single value repeats itself in the sequence, the whole sequence will repeat itself. Hence, in our case we obtained a sequence with 10 distinct values and then the sequence starts to repeat itself.

Figure 6.1: Plot of the Linear Congruential generator example



Because of the effectiveness and simplicity of the LCG, they are the most widely used in practice. The mixed Linear Congruential generator was first proposed by Lehmer [1951] [29] and takes the form of,

$$a_{i+1} = (ka_i + c) mod(m)$$

$$u_{i+1} = \frac{a_{i+1}}{m}$$

This equation is called the mixed Linear Congruential generator and the example in the previous case is called the pure Linear Congruential generator. Just as we discussed in the previous case, k and m should be integers. Now we added an extra parameter c, which should also be an integer.

This type of algorithm has undergone some extensive research and a lot is known about the set of values that is generated, $[u_1, u_2, ..., u_n]$. There are

some set out conditions that specify the choice of parameters so that the generator has a full period. A full period will mean the generation of $(m-1)$ distinct set of values, generated from the seed value x_0. These parameter choices are set out in Knuth [1998] and are as follows:

1. c and m should be chosen as prime numbers

2. Every prime number that divides m should divide $a-1$ as well.

3. $a-1$ is divisible by 4 if m is.

Due to the choice of parameters as specified above, we now have that if m is a power of 2, the generator will have a period of $m-1$ if c is chosen to be odd, and $a = 4n+1$ for some integer n. As a consequence, if we have chosen that $c = 0$ and m represents a prime number, then a period of $m-1$ is achieved, with any particular seed value $x_0 \neq 0$. This only holds if the following two conditions are met;

1. $a^{m-1} - 1$ is a multiple of m

2. $a^j - 1$ is not a multiple of m for j=1,2,3,..,m-1.

A specific number a that satisfies both abovementioned properties is called a primitive root of m. We then observe that the sequence becomes

$$x_0, ax_0, a^2x_0, ...mod(m)$$

for a chosen $c = 0$.

Example 6.2.2. Let us consider the following LCG;

$$a_{i+1} = (k\, a_i \, + c)\, mod(m)$$

$$u_{i+1} = \frac{a_{i+1}}{m}$$

where:

k = 11

m = 71

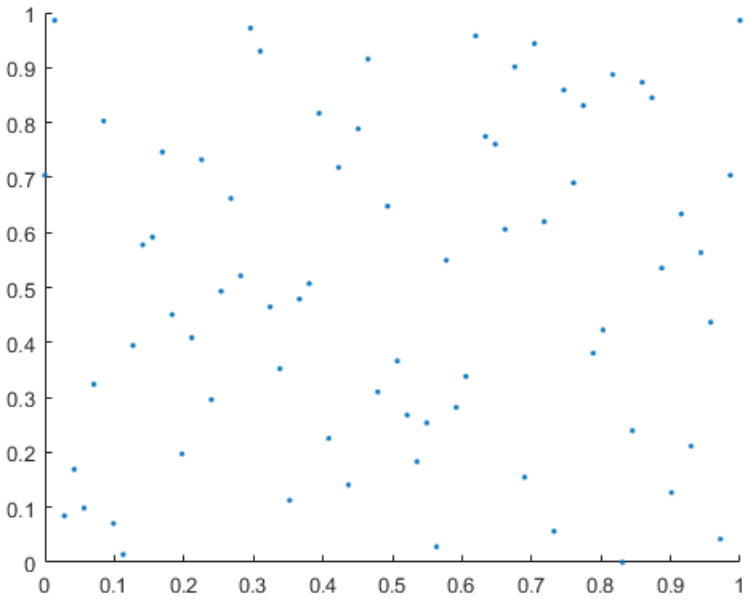
$a_0 = 3$

c = 17

When we apply this basic algorithm recursively, we get to the following sequence;
$[3, 50, 70, 6, 12, 7, 23, 57, 5, 1, 28, 41, 42, 53, 32, 14, 29, 52, 21, 35, 47, 37, 69, 66, 33, 25, 8, 34, 36, 58, 16, 51, 10, 56, 65, 22, 46, 26, 19, 13, 18, 2, 39, 20, 24, 68, 55, 54, 43, 64, 11, 67, 44, 4, 61, 49, 59, 27, 30, 63, 0, 17, 62, 60, 38, 9, 45, 15, 40, 31, 3^*, 50^*, 70^*]$

We observe that once a single value repeats itself in the sequence, the whole sequence will repeat itself. Hence, in our case we obtained a sequence with 70 distinct values and then the sequence starts to repeat itself. Where the sequence starts to repeat itself is marked with *. This shows, with the careful consideration of the choosing of parameters, it will significantly increase the efficiency of the Linear Congruential generator to produce more distinct values.

Figure 6.2: Plot of the mixed Linear Congruential generator example



Hence, to find the maximum period length that can be possibly produced by such a Linear congruential generator is thoroughly discussed in Lewis, Goodman, and Miller[1969], L'Ecuyer[1988], Fishman and Moore[1986], and Park and Miller[1988].

In all the abovementioned research, the modulus m is a large prime that does not exceed the value of $2^{31} - 1$. This represents the largest integer that can be represented by a 32-bit word. This number also happens to be a prime number and is known as the Mersenne prime.

Table 1 represents previously done research on Linear Congruential generators. This research indicated that using a Linear Congruential generator can produce over 2 billion distinct values, when parameter selection is very carefully considered. For practical application these "extremely" large generators are not very useful. Since they take a lot of computational time to

Table 6.1: List of previous LCG research

Modulus m	Multiplier k	Reference
2147483647 $(2^{31}-1)$	16807	Lewis, Goodman, and Miller[1969]
2147483647 $(2^{31}-1)$	39373	L'Ecuyer[1988]
2147483647 $(2^{31}-1)$	742938285	Fishman and Moore[1986]
2147483647 $(2^{31}-1)$	1226874159	Fishman and Moore[1986]
2147483399	40692	L'Ecuyer[1988]
2147483563	40014	L'Ecuyer[1988]

create the sequence and requires a significant amount of memory to process the algorithm.

6.3 General sampling methods

We will briefly discuss the principal of sampling methods. We already discussed the generation of random sequences and we made the assumption that it is possible to generate the ideal sequence of random numbers to use for our application procedures. From [36], we have that a sequence is generated $U_1, U_2, ..., U_n$, is available, and conforms to the following distribution:

$$P(U_i \leq u) = \begin{cases} 0 & ; u < 0 \\ u & ; 0 \leq u \leq 1 \\ 1 & ; u > 1 \end{cases}$$

where each is uniformly distributed between $[0, 1]$.
We now use our uniformly generated sequence and we need to transform them into sample paths of stochastic processes, with specific distributional properties. Researched has been conducted extensively on both general purpose methods and specialized algorithms for specific cases. In the subsections that follows, we discuss the two most widely known techniques, namely, the Inverse transform method and Acceptance-rejection method.

6.3.1 Inverse transform method

If we want to sample from a cumulative distribution function F, then we want to generate a random variable X, that conforms to the following property $P(X \leq x) = F(x)\, \forall x$. So, the Inverse transform method can be applied in the following way:

$$X \ = F^{-1}(U)\, where U \ uniform[0,1] \qquad \text{(eq:6.3.1)}$$

where the F^{-1} represents the inverse function of F. For the Inverse transform method to work efficiently, it is important that F is well defined so that F is strictly increasing, otherwise we need to implement a rule to break ties. For example [36],

$$F^{-1}(u) \ = \ inf[x : F(x) \geq u]$$

if we have that for many different x values for which $F(x) = u$, the specified rule above will choose the smallest value.

If the function F is defined in such a way that the function has flat sections or solely consists of a flat section, the these flat sections of F then correspond to intervals of zero probability for the random variable that was generated. Otherwise if F is well defined as a strictly increasing function, then anywhere the density will be nonzero anywhere.

To validate our statement as seen in eq:7.3.1, we verify that the Inverse transform method generates samples from F, we then need to check the distribution of the X that the method produces:

$$\begin{aligned} F_U(x) \ = \ P(X \leq x) &= P(F^{-1}(U) \leq x) \\ &= P(U \leq F(x)) \\ &= F(x) \end{aligned}$$

since $P(U \leq F(x))$ is defined as the cumulative function $F(x)$ and F^{-1} is defined to hold the following properties $\{F^{-1}(u) \leq x\}$ and $\{u \leq F(x)\}$, which holds ideally when F is a continuous function and holds for all u and x.

A more general procedure for generating a random number from a distribution F_X applying the Inverse transform method is as follows:

* Find the formula for the quantile function F_X^{-1}

* Generate a uniform random number u.

* Return the random number $x = F_X^{-1}(u)$

We now consider two examples to illustrate the Inverse transform method. We consider some modifications to the Inverse transform method to increase the efficiency with which it is implemented.

Example 6.3.1. Let us consider a Exponential distribution with mean θ and has a cumulative distribution function F as follows:

$$F(x) = 1 - e^{-\frac{x}{\theta}} \quad , for\ x \geq 0.$$

This distribution represents time between jumps of a Poisson process with rate $\frac{1}{\theta}$. Inverting this exponential cumulative distribution function yields:

$$\begin{aligned} 1 - U &= e^{\frac{-x}{\theta}} \\ ln(1-U) &= \frac{-x}{\theta} \\ \Rightarrow X &= -\theta \ln(1-U) \\ \Rightarrow &= -\theta \ln(U) \end{aligned}$$

When we implement the inverse transform method it is unlikely to be the fastest method for sampling from a certain distribution. Nonetheless, the

inverse transform method has some very attractive qualities. Firstly, the inverse transform method plots U one-to-one to the output X, if we have that the function F is strictly increasing. This can be utilized in variance reduction techniques (antithetic variates, control variates, etc.) and sensitivity analysis. Hence, the inverse transform method requires only one uniformly distributed random variable to generate a sample.
Glasserman [36] states "This is particularly important in using Quasi-Monte Carlo methods where the dimension of a problem is often equal to the number of uniforms needed to generate one path. Methods that require multiple uniforms per variable generated result in higher-dimensional representations for which Quasi-Monte Carlo may be much less effective."

6.3.2 Acceptance-Rejection method

The acceptance-rejection method was first introduced by Von Neumann[1951] [34] and for the explanation of the Acceptance rejection method, we base the following on the lecture notes of Karl Sigman[2007] [32] and Paul Glasserman [36].
As we have already discussed, we are able to find an explicit expression for the inverse cumulative distribution function, denoted by $F^{-1}(x)$ and we aim to generate $F(y) = P(Y \leq y)$, but this is not always possible. There may exist more efficient sampling methods other than the Inverse transform method.
Firstly, we assume that the F that we are trying to generate has a probability density function $f(x)$ and the function is continuous. We then use a more convenient distribution, say G, which is closely related to our target distribution F. We assume that both F and G have respective probability distribution functions f and g. We then reject a subset of generated candidates that do not conform to our target distribution. This sampling technique does not only apply to univariate cases but also extends to multivariate cases.
We assume that the ratio $\frac{f(x)}{g(x)}$ is bounded by a constant $c > 0$, $sup_x\{\frac{f(x)}{g(x)}\} \leq c$.

So then, suppose that we have a density function f defined on some set X and let g also be a density on X, for which we know how to generate samples from. Then the following property holds,

$$f(x) \leq cg(x),\ \forall\, x \in X$$

for some bounded constant c. This ratio is bounded between $(0, 1]$,

$$0 < \frac{f(x)}{cg(x)} \leq 1.$$

Now we will discuss the general algorithm for the Acceptance-Rejection method in the case of continuous random variables.

1. Generate a random number y from the distribution g.

2. Generate a random number u from $Uniform(0, 1)$.

3. If $u \leq \frac{f(y)}{cg(y)}$, return y. Otherwise, go back to step 1.

Theorem 4. *Proving the Acceptance-Rejection method. We have to show that the conditional distribution of Y given that $U \leq \frac{f(Y)}{cg(Y)}$, is indeed the targeted distribution F. Sigman [32]. So, $P(Y \leq y | U \leq \frac{f(Y)}{cg(Y)}) = F(y)$*

Proof, Firstly, $P(U \leq \frac{f(Y)}{cg(Y)} | Y \leq y) = \frac{f(Y)}{cg(Y)}$ and because Y has the density g(y), we have

$$\begin{aligned} P(U \leq \frac{f(Y)}{cg(Y)}) &= \int_{-\infty}^{\infty} \frac{f(y)}{cg(y)} g(y)\, dy \\ &= \frac{1}{c} \int_{-\infty}^{\infty} f(y) dy \\ &= \frac{1}{c} \\ &= p \end{aligned}$$

Then we use the following computations:

$$
\begin{aligned}
P(U \leq \frac{f(Y)}{cg(Y)} | Y \leq y)\, \frac{G(y)}{p} &= \frac{P(U \leq \frac{f(Y)}{cg(Y)}, Y \leq y)}{G(y)}\, \frac{G(y)}{p} \\
&= \int_{-\infty}^{y} \frac{P(U \leq \frac{f(y)}{cg(y)}) | Y = z \leq y}{G(y)} g(z)dz\, \frac{G(y)}{p} \\
&= \frac{1}{G(y)} \int_{-\infty}^{y} \frac{f(z)}{cg(z)} g(z)dz\, \frac{G(y)}{p} \\
&= \frac{1}{cG(y)} \int_{-\infty}^{y} f(z)\, dz\, \frac{G(y)}{p} \\
&= \frac{F(y)}{cG(y)}\, \frac{G(y)}{\frac{1}{c}} \\
&= F(y) \qquad \square
\end{aligned}
$$

We can implement the following algorithm to generate for the normal distribution, $X \sim N(\mu, \sigma^2)$ and express the distribution as $X = \sigma Z + \mu$. Z denotes the standard normal distribution and our $g(x)$ is chosen as the exponential distribution with rate 1, *i.e* $g(x) = e^{-x},\ x \geq 0$

1. We generate Y with the exponential distribution at rate 1. So, generate U and set $Y = -ln(U)$

2. Generate U

3. If $U \leq e^{\frac{-(Y-1)^2}{2}}$, set $|Z| = Y$ otherwise return to 1.

4. Generate U and if $U \leq 0.5$ then set $Z = |Z|$ and set $Z = -|Z|$ if $U > 0.5$.

6.3.3 Box-Muller method

The Box-Muller method was first introduced by George Edward Pelham Box and Mervin Edgar Muller in 1958. The Box-Muller method is a pseudo-

random number sampling method for generating pairs of independent, standard, normally distributed random numbers.
The Box-Muller method is a brilliant trick by producing two independent standard normals from two independent uniforms. It is based on a very familiar trick using polar coordinates as shown in J. Goodman [33]. Firstly we investigate the following integral,

$$I = \int_{-\infty}^{\infty} e^{\frac{-x^2}{2}}\, dx$$

This integral cannot be calculated by "integration". The indefinite integral does not have an algebraic expression in terms of elementary functions such as, exponentials, logarithms, trig functions, etc. However, we can manipulate the following integral,

$$\begin{aligned} I^2 &= \int_{-\infty}^{\infty} e^{\frac{-x^2}{2}}\, dx \int_{-\infty}^{\infty} e^{\frac{-y^2}{2}}\, dy \\ &= \iint_{-\infty}^{\infty} e^{\frac{-(x^2+y^2)}{2}}\, dxdy \end{aligned}$$

Now, the last integral can be calculated in terms of polar coordinates. Where $x = r\cos(\theta),\ y = r\sin(\theta)$ and the area element will become $dxdy = rdrd\theta$,

$$\begin{aligned} I^2 &= \iint_{\theta=0}^{2\pi} e^{\frac{-r^2}{2}} r dr d\theta \\ &= 2\pi \int_{\theta=0}^{2\pi} e^{\frac{-r^2}{2}} r dr \\ &= 2\pi \int_{s=0}^{\infty} e^{-s} ds \qquad\qquad (substituting\ s = \frac{r^2}{2}) \\ &= 2\pi \end{aligned}$$

This integral was calculated by making the substitution of $s = \frac{r^2}{2}$, which

yields $ds = rdr$.

The Box-Muller algorithm is a probabilistic interpretation of the above-mentioned manipulation. So, if we look at the problem where (X, Y) is represented by a pair of standard normals, then the probability density is represented by the following product,

$$\begin{aligned} f(x,y) &= \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \,.\, \frac{1}{\sqrt{2\pi}} e^{\frac{-y^2}{2}} \\ &= \frac{1}{2\pi} e^{\frac{-(x^2+y^2)}{2}} \end{aligned}$$

Since, the density is symmetric and we now consider the polar coordinates random variables (R, Θ) defined by $0 \leq \Theta < 2\pi$ and $X = R\cos(\Theta)$ and $Y = R\sin(\Theta)$. Hence, we have that Θ is uniformly distributed over the interval $[0, 2\pi]$ and is sampled using,

$$\Theta = 2\pi U_1$$

Unlike the previous distribution, the distribution for R is expressed as follows,

$$\begin{aligned} F(R) &= P(R \leq r) \\ &= \int_{r'=0}^{r} \int_{\Theta=0}^{2\pi} \frac{1}{2\pi} e^{\frac{-r'^2}{2}} r dr d\theta \\ &= \int_{r'=0}^{r} e^{\frac{-r'^2}{2}} r dr \end{aligned}$$

Now, if we apply the same substitution as mentioned above $s = \frac{r'^2}{2}$, we obtain

$$\begin{aligned} G(r) &= \int_{s=0}^{\frac{r^2}{2}} e^s ds \\ &= 1 - e^{\frac{-r^2}{2}} \end{aligned}$$

Hence, we can sample R by solving the distribution function.

$$\begin{aligned} G(R) &= 1 - e^{\frac{-R^2}{2}} \\ &= 1 - U_2 \qquad\qquad (Recall\ 1 - U_2\ is\ standard\ uniform\ if\ U_2\ is.) \end{aligned}$$

We can get to a solution,

$$\begin{aligned} X &= \sqrt{-2\ln U_2}\cos(2\pi U_1) \\ Y &= \sqrt{-2\ln U_2}\sin(2\pi U_1) \end{aligned}$$

This X and Y will produce independent standard normals using independent uniform variables U_1 and U_2.

6.4 Variance reduction

Monte Carlo integration has an error variance of $\frac{\sigma^2}{n}$. This shows that we are able to obtain better sampling by increasing the number of simulations n. There is, however, a trade-off. The more simulations we generate, the more computational time is used to generate these samples. So, the trade-off is between accuracy and computational time. Hence, we rather use methods to reduce our σ instead. These methods that we use to reduce our σ is called Variance reduction techniques.
These methods will increase the efficiency of Monte Carlo [36] simulation by reducing the variance of the related simulation estimates. These methods call on two broad strategies for reducing the variance. Firstly, taking advantage of the tractable features of a specific model to correct simulation outputs, and reducing the variability of the simulation inputs. We will discuss Control variates, Antithetic variates, and Importance sampling.

6.4.1 Antithetic variates

Antithetic variates, according to Glasserman [36] is a method that uses negative dependence between pairs of replications to reduce variance. The Antithetic variance method takes on various forms and the most broadly use is if U is uniformly distributed over the unit interval $[0, 1]$. We have the property that if this is true that U is uniformly distributed over $[0, 1]$ then $1 - U$ will also be uniformly distributed. Consequently, we are able to generate a sequence of random uniformly distributed variables $U_1, U_2, ..., U_n$ and we are able to generate a second path $1 - U_1, 1 - U_2, ..., 1 - U_n$ using the same law of the simulated process. Hence, for every simulated value U_i and $1 - U_i$, which forms an Antithetic pair of variables, is accompanied each by a small and large value. For instance, if U_1 is simulated to be a "large" value (> 0.5) the accompanied antithetic will be a relatively smaller value (< 0.5) since the variables are uniformly distributed and bounded over the unit interval. This suggests that if an unusually large or small number is generated it will be balanced by its antithetic pair, which will in return reduce the variance of the simulated process.

We are able to use the inverse transform method to extend this method to other distributions. If, under the assumption that F^{-1} is monotone[1], we have that $F^{-1}(U)$ and $F^{-1}(1 - U)$ both have the distribution of F but are antithetic to each other. In the case where we have a symmetric distribution, that is symmetric around the origin, such as the normal distribution, then we have that $F^{-1}(u)$ and $F^{-1}(1 - u)$ will have the same values but will have opposite signs.
In a more practical application sense, where we need to simulate independent standard normal random variables $N(0, 1)$, we are able to pair the sequences of the simulated antithetic variates. Hence, we pair $Z_1, Z_2, Z_3, ...$ which are

[1]A function F is monotone on an interval [a,b] if F is either increasing on the whole of [a,b] or else decreasing on [a,b].

independent and identically distributed $N(0,1)$ variables with $-Z_1, -Z_2, -Z_3, ...$ which are also independent and identically distributed $N(0,1)$ variables.
If we use these pairs of standard normally distributed random variables to simulate the increments of a Brownian path and we compare Z_i and $-Z_i$, we observe that $-Z_i$ is a reflection of Z_i around the origin. We expect that these pairs of Antithetic variables, which incorporates the use of the reflection of Z_i will lower the variance of the estimated process.

If we wish to analyze the approach more in depth, assume that we want to estimate $E[X]$ with the use of antithetic sampling sequence of pairs of observations $(X_1, \tilde{X}_1), (X_2, \tilde{X}_2), (X_3, \tilde{X}_3), ..., (X_k, \tilde{X}_k)$. There are two specific properties of the simulated Antithetic pair sequence:

1. The simulated pairs $(X_1, \tilde{X}_1), (X_2, \tilde{X}_2), (X_3, \tilde{X}_3), ..., (X_k, \tilde{X}_k)$ are independent and identically distributed,

2. $\forall\, k$, X_k and $\tilde{X}_k$ will have the same distribution but the pair will not be independent.

For illustration purposes we use X to indicate a random variable with the distribution of the Antithetic pair X_k and $\tilde{X}_k$.
The estimator for these antithetic sequence pair will be the average for all of the $2n$ observations that were simulated and can be represented as,

$$\begin{aligned}\tilde{X}_{est} &= \frac{1}{2n}(\sum_{i=1}^{k} X_i \; + \; \sum_{i=1}^{k} \tilde{X}_i) \\ &= \frac{1}{n}\sum_{i=1}^{k}(\frac{X_i + \tilde{X}_i}{2})\end{aligned}$$

Hence, we have that:

$$(\frac{X_i + \tilde{X}_i}{2}) \qquad ;\forall\, i = 1, 2, 3, ..., k$$

represents the average between the antithetic pairs and we can clearly deduce that X_{est} is the sample mean of n independent observations

$$(\frac{X_1, \tilde{X}_1}{2}), (\frac{X_2, \tilde{X}_2}{2}), (\frac{X_3, \tilde{X}_3}{2}), ..., (\frac{X_k, \tilde{X}_k}{2}) \qquad \text{(eq:6.4.1)}$$

we have that the central limit theorem also applies and yields,

$$\frac{\tilde{X}_{est} - E[X]}{\frac{\sigma_{est}}{\sqrt{k}}} \sim N(0, 1)$$

for k sufficiently large and σ_{est} is presented as follows,

$$\sigma_{est} = Var[\frac{X_i + \tilde{X}_i}{2}]$$

We now replace σ_{est} with s_{est}, which represents the sample standard deviation for the k generated values as seen in eq:7.4.1. Hence, the limit of the distribution will continue to hold with this replacement. This now provides a asymptotic justification for a $(1 - \alpha)$ confidence interval and is represented as,

$$\tilde{X}_{est} \pm z_{\frac{\alpha}{2}} \frac{s_{est}}{\sqrt{k}}$$

A comparison needs to be made between the standard Monte Carlo estimator and the antithetic variates, and then the use of antithetic variates would be beneficial in terms of computational cost and accuracy. To be able to make a fair comparison between the two, we observe that the antithetic method will produce twice the number of variables and, hence, we deduce that twice the computational effort will be required to generate these pairs [36]. Thus, we will ignore any potential computational savings from, for example, flipping the signs from the previously generated $Z_1, Z_2, Z_3, ...$ rather than generating

new normal variables. This is only appropriate when the computational cost of generating these inputs is a small fraction of the total cost of simulating Y_i.

So, if we work under this assumption, the effort that is necessary to calculate $\tilde{X}_{est}$ is approximately equal computing the sample mean of $2n$ independent replications. We can then make a fair comparison of the variances of the two relevant estimators. Thus antithetics reduce the variance if,

$$Var(\tilde{X}_{est}) \ < \ Var(\frac{1}{2n}\sum_{i=1}^{2n} X_i)$$

so,

$$Var(\frac{X_i \ + \ \tilde{X}_i}{2}) < Var(X_i)$$

or,

$$Var(X_i \ + \ \tilde{X}_i) < 2Var(X_i)$$

We then expand the variance for antithetics with some statistical properties,

$$\begin{aligned} Var(X_i \ + \ \tilde{X}_i) &= Var(X_i) + 2Cov(X_i, \tilde{X}_i) + Var(\tilde{X}_i) \\ &= 2Var(X_i) + 2Cov(X_i, \tilde{X}_i) \end{aligned}$$

since X_i and $\tilde{X}_i$ will have the same variance, because they originate from the same distribution. Hence, the condition for Antithetic variates should comply with in order to reduce the variance of the estimator, the following condition should hold,

$$Cov(X_i, \tilde{X}_i) < 0$$

This condition can easily be demonstrated [35]. We define ϕ so that $C_i = \phi(Z_i)$. C_i is the unbiased estimator of an option and ϕ is the composition

of the mappings from Z_i to the stock price and from the stock price to the discounted payoff. The stock price process is given as,

$$S_t^{(i)} = S_0 e^{((r-\frac{\sigma^2}{2})T-\sigma\sqrt{T}Z_i)} \qquad ;\forall i = 1, 2, ..., k$$

As this is the composition of two increasing functions. We have that ϕ is monotone, so by the standard inequality of Barlow and Proschan, 1975.

$$\begin{aligned} E[\phi(Z_i)\phi(-Z_i)] &\leq E[\phi(Z_i)]E[\phi(-Z_i)] \\ E[\phi(Z_i)\phi(-Z_i)] - E[\phi(Z_i)]E[\phi(-Z_i)] &\leq 0 \\ \Rightarrow Cov(C_i, \tilde{C}_i) &\leq 0 \end{aligned}$$

Hence, we may conclude that antithetics will reduce the variance and will be beneficial to be implemented into Monte Carlo. The method of antithetic variates increases efficiency when it comes to the pricing of European options and other options that depend on monotonically inputs (such as Asian options). Other options that depend less on monotonically inputs, such as Barrier options ,suggests that the use of the method of antithetic variates may be less effective.

When we are interested in calculating the confidence intervals, implementing the method of antithetic variates, it is of utmost importance that the standard error should be calculated using the sample standard deviation of only the n averaged pairs $(C_i + \tilde{C}_i)/2$ and not the $2n$ individual observations $C_1, \tilde{C}_1, C_2,$
$\tilde{C}_2, ..., C_k, \tilde{C}_k$.

The pairs, for example, $C_1, \tilde{C}_1$ are not independent of each other but, they are dependent between pairs of simulated results.

Figure 6.3: Plot of 1000 step Geometric Brownian motion with the Antithetic variates.

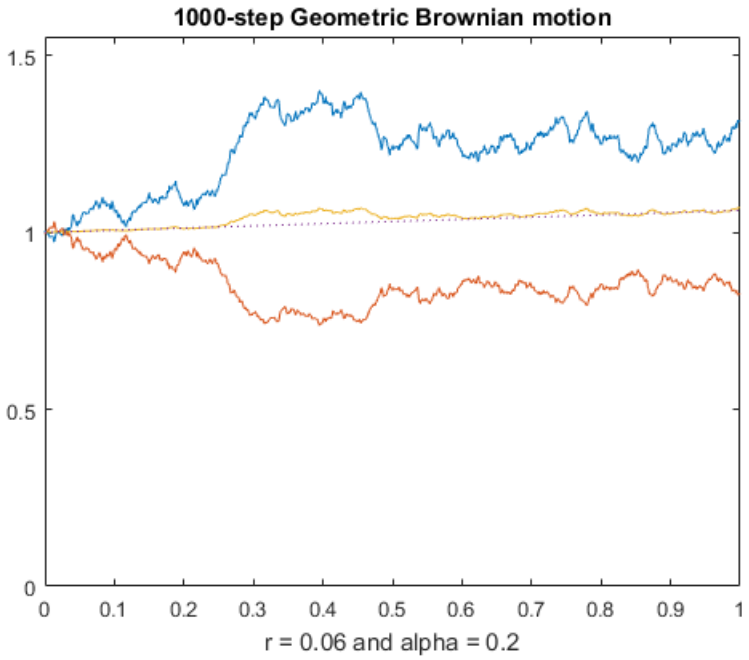


Figure 6.3 represents a 1000 step geometric Brownian motion, which is depicted in blue. We then have its antithetic counterpart which is depicted in orange and the mean is represented by a dotted line. The average at any certain point is represents as follows $(C_i + \tilde{C}_i)/2$.

6.4.2 Control variates

Control variates is one of the most applicable and effective variance reduction methods that are applied to the Monte Carlo simulation, to improve the process efficiency [36]. The Control variates method exploits information about known error estimates to reduce the error in an estimate of an unknown quantity.

To be able to describe the method, we let $Y_1, Y_2, Y_3, ..., Y_n$ be outputs from n replications of a simulation. In our case we want Y_i to represent the discounted payoff of a derivative on the i'th simulated path. We assume that Y_i is independently and identically distributed and our objective is to find an estimate for $E[Y_i]$. The same as before, our estimator for the sample mean is

$$\bar{Y} = \frac{(Y_1 + Y_2 + Y_3 + ... + Y_n)}{n}.$$

Hence, this represents an unbiased estimator that converges to probability 1 as $n \to \infty$.

If we suppose that we calculate another output with each replication, say X_i, in conjunction with the Y_i output variable. We also assume that the generated pairs (X_i, Y_i), $\forall\, i = 1, 2, 3, ..., k$, are independent and identically distributed and the fact that $E[X]$ is known from the simulation of X_i. In statistical terms we denote (X, Y) as the pair of random variables with the same distribution as each individual pair (X_i, Y_i). Then for any constant b we are able to apply the following procedure in the effort to decrease variance of the simulated process,

$$Y_i(b) = Y_i - b(X_i - E[X])$$

The expression above is for the i'th replication and then we are able to approximate the sample mean with the following,

$$\begin{aligned} \bar{Y}(b) &= \frac{1}{n}\sum_{i=1}^{n}(Y_i - b(X_i - E[X])) \\ &= \bar{Y} - b(\bar{X} - E[X]) \end{aligned}$$

This expression is called a Control variate estimator. The error that is obtained with $(X_i - E[X])$ serves as a control in the approximation of $E[Y]$. If we analyze this Control variate expression, we see that both are an unbiased

and consistent estimator for the approximation of $E[Y]$. The expression is unbiased because

$$\begin{aligned} E[\bar{Y}(b)] &= E[\frac{1}{n}\sum_{i=1}^{n}(Y_i - b(X_i - E[X]))] \\ &= E[\bar{Y} - b(\bar{X} - E[X])] \\ &= E[\bar{Y}] - b(E[\bar{X}] - E[X]) \\ &= E[\bar{Y}] \\ &= E[Y] \end{aligned}$$

And, to prove that the expression is consistent, we have that,

$$\begin{aligned} \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\bar{Y}_i(b) &= \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}(Y_i - b(X_i - E[X])) \\ &= E[Y - b(X - E[X])] \\ &= E[Y] - b(E[X] - E[X]) \\ &= E[Y] \end{aligned}$$

It is also possible to find a expression for the variance of the Control variate formulation,

$$\begin{aligned} Var(Y_i(b)) &= Var(Y_i - b(X_i - E[X])) \\ &= \sigma_Y^2 - 2b\sigma_X\sigma_Y\rho_{XY} + b^2\sigma_X^2 \\ &\equiv \sigma^2(b) \end{aligned} \qquad \text{(eq:6.4.2)}$$

where σ_Y^2 and σ_Y respectively represent the variance and standard deviation of Y(where Y represents some discounted payoff of a derivative). ρ_{XY} represents the correlation between the approximation we aim to obtain,Y, and the Control variate X. The Control variate estimator $\bar{Y}(b)$ has variance of $\frac{\sigma^2(b)}{n}$ and the sample mean has a variance $\frac{\sigma_Y^2}{n}$. Thus we deduce that the control

variate estimator has a smaller variance then the standard estimator if,

$$b^2\sigma_X < 2b\sigma_Y\rho_{xy}.$$

Hence, the choice of b is very important. We choose b in such a way that the optimal coefficient b will minimize the variance, as seen in eq:6.4.2. This optimal choice is given by,

$$\begin{aligned} b &= \frac{\sigma_Y}{\sigma_X}\rho_{XY} \\ &= \frac{\sigma_Y}{\sigma_X}\frac{Cov(X,Y)}{\sigma_X\sigma_Y} \\ &= \frac{Cov(X,Y)}{Var(X)} \end{aligned}$$

As shown in the Glasserman [36] textbook, we are able to substitute this choice of b back into eq:6.4.2 and simplify. We are then able to obtain the ratio of the variance of the optimally controlled variable to that of the variance of the uncontrolled variable. This ratio simplifies to,

$$\frac{Var(\bar{Y} - b(\bar{X} - E[X]))}{Var(\bar{Y})} = 1 - \rho_{XY}^2 \qquad \text{(eq:6.4.3)}$$

From Glasserman [36] there are a few observations that can be made about the above mentioned ratio,

1. With the specification of b as the optimal coefficient, the effectiveness of this choice is represented in the correlation between the estimation of interest Y and the control generated estimate X. This is measured from eq:7.4.3. as the variance reduction ratio. The associated sign that is calculated with any optimal coefficient is irrelevant since the effect is absorbed by the optimal coefficient b.

2. If we work under the assumption that each replication requires the same (more or less) computational effort, the eq:6.4.3. measures the com-

putational speed up when making use of a control variate. Hence, the number of replications that is required to simulate Y_i abd achieve the same variance as n replications of the control variate is given by $\frac{n}{(1-\rho_{XY}^2)}$.

3. If we the consider the variance reduction factor of $\frac{1}{(1-\rho_{XY}^2)}$. We notice that this ratio increases very sharply as $|\rho_{XY}|$ approaches 1 and on the other hand the ratio decreases very sharply as $|\rho_{XY}|$ moves away from 1. Hence, the variance reduction factor suggest that strong correlation should exist between the estimate of interest Y and the control variate X, for the Control variate method to yield substantial benefits.

One further remark that one should keep in mind, that the optimal coefficient will not always be known in practice. If for example, $E[Y]$ is unknown in practice it is very unlikely that either σ_Y or ρ_{XY} will be known quantities. However, it is most likely that one will reap the most benefit of using the b as it is specified above. Now we suggest that we replace our population parameters with their sample counterparts. Hence, we obtain the following expression with sample parameters,

$$\hat{b} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

The Control variate method can also be extended to the use of multiple controls. The use of multiple controls can be justified when the simulation involves n underlying assets or we want to use multiple options as controls with multiple strikes and maturities. This is, however, beyond the scope of this paper. Our aim is to improve the Longstaff and Schwartz approach with single control variates.

6.4.3 Importance Sampling

Importance sampling is a variance reduction technique in the sense that it tries to minimize variance by changing the probability measure from which

paths are generated [36]. Importance sampling is a standard variance reduction technique and is well known in financial mathematics. For example, if we are able to switch between the objective probability measure to the risk neutral probability measure to enable us to find a better interpreted and calculated version of the expected value we aim to find. Hence, with the implementation of important sampling we aim to allocate more weight to more important outcomes and thus increasing the efficiency.
To better illustrate this technique, let us consider the following problem;

$$\begin{aligned}\alpha &= E[g(Y)] \\ &= \int g(y)f(y)dy\end{aligned}$$

where we define our X as a random element of $\mathbb{R}^d$ with probability density f [36]. We define g to be a function from $\mathbb{R}^d \to \mathbb{R}$. The ordinary Monte Carlo estimator is still defined as,

$$\begin{aligned}\hat{\alpha} &= \hat{\alpha}(n) \\ &= \frac{1}{n}\sum_{i=1}^{n} g(Y_i)\end{aligned}$$

where the sequence $Y_1, Y_2, ..., Y_n$ represent independent draws from f. Now, we let h be another probability density that is defined on $\mathbb{R}^d$ and has to satisfy the following inequality,

$$f(y) > 0 \Rightarrow h(y) > 0 \qquad ; \forall\, y \in \mathbb{R}^d$$

Now, we are able to rewrite α in the following way;

$$\alpha = \int \frac{f(y)}{h(y)} g(y)h(y)dy$$

we can then interpret the integral as the expectation of the density with

respect to h and we therefore find the following expression,

$$\alpha = \tilde{E}[g(Y)\frac{f(Y)}{h(Y)}] \tag{eq:6.4.4}$$

where $\tilde{E}$ indicates the expectation is taken with Y distributed according to h. If the sequence $Y_1, Y_2, Y_3, ..., Y_n$ indicates independent draws from g, the importance sampling estimator is represented by the following expression,

$$\begin{aligned} \hat{\alpha}_g &= \hat{\alpha}_g(n) \\ &= \frac{1}{n}\sum_{i=1}^{n} g(Y_i)\frac{f(Y_i)}{h(Y_i)} \end{aligned} \tag{eq:6.4.5}$$

The value that is computed by $\frac{f(Y_i)}{h(Y_i)}$ is represented as a weight and is the likelihood ratio at any Y_i.

From eq:6.4.4. we are able to see that $E[\tilde{\alpha}_g] = \alpha$, which proves that the importance sampling estimator is unbiased. Thus $\tilde{\alpha}_g$ is an unbiased estimator for α. If we want to compare the variance obtained with importance sampling and compare it to the normal variance obtained, then it would be sufficient to only consider the second moments only. The expression for considering the second moment of importance sampling is as follows,

$$\tilde{E}[(g(Y)\frac{f(Y)}{h(Y)})^2] = E[(g(Y)^2\frac{f(Y)}{h(Y)})]$$

The second moment without importance sampling is $E[(g(Y))^2]$. Hence, the expectation with importance sampling can be either larger or smaller than the expectation without importance sampling. The choice of h is very important [36]. A certain choice of h may have a significant influence on the expectation with importance sampling. A choice of an effective h may lead to very effective importance sampling. If we assume that g is non-negative, the product of $g(y)f(y)$ will also be non-negative and it is possible to be

normalized to the following probability density function.

$$h(y) \propto g(y)f(y) \tag{eq:6.4.6}$$

By rearranging eq:6.4.6 we obtain that $(g(Y)\frac{f(Y)}{h(Y)})$ equals a constant proportionality regardless the value of Y_i.
An important quote follows directly from Glasserman [36] states "In designing an effective importance sampling strategy, we should try to sample in proportion to the product of g and f. In option pricing applications, g is typically a discounted payoff and f is the risk-neutral density of a discrete path of underlying assets. In this case, the importance of a path is measured by the product of its discounted payoff and its probability density."

Likelihood Ratios

If we consider option pricing applications with respect to importance sampling, we may assume that Y represents discrete paths for the underlying assets. Let us then consider the discrete path for the stock price denoted as $S_{t_j}, for\ j = 0, 1, 2, 3, ..., n$ and we make the assumption that the process is Markov. Then suppose the conditional distribution of S_{t_j} given $S_{t_j} - 1 = y$, has a density of $f(y, .)$. As importance sampling dictates, we now consider a change in measure. Transition densities f_i are replaced with new transition densities h_j. Hence, we deduce the likelihood ratio for this change in measure is as follows,

$$\prod_{j=1}^{n} \frac{f_j(S_{t_{j-1}}, S_{t_j})}{h_j(S_{t_{j-1}}, S_{t_j})}$$

If we make our expectation more exact, so that if E is the expectation under the original measure and $\tilde{E}$ denotes the expectation under the new measure. Then we have the following expression,

$$E[g(S(t_1), S(t_2), ..., S(t_n))] = \tilde{E}[g(S(t_1), S(t_2), ..., S(t_n)) \prod_{j=1}^{n} \frac{f_j(S_{t_{j-1}}, S_{t_j})}{h_j(S_{t_{j-1}}, S_{t_j})}] \quad \text{(eq:6.4.7)}$$

All functions of g, under the original measure, the expectations exists and is finite. In our stock price we assume that S_{t_0} is a constant and is represented by density f_0 under the original measure and is represented by density h_0 under the new measure. This has the implication that we obtain an extra additional initial factor of $\frac{f_0(S_{t_0})}{h_0(S_{t_0})}$ in the likelihood product.
We usually simulate a stock price path through recursion that is driven by independent and identically distributed random vectors $Y_1, Y_1, ..., Y_n$. We are now able to denote in the form of,

$$S_{t_{j+1}} = H(S_{t_j}, Y_{j+1})$$

Y_i is very likely to be normally distributed. But we need to assume that for all Y_i have the same common density f. If we then apply the change in measure from the original measure to the new measure, the new common density changes to g. If a change of measure takes place, we assume that the new measure will preserve the independence of Y_i. The likelihood then changes as follows,

$$\prod_{j=1}^{n} \frac{f(Y_i)}{h(Y_i)}$$

Hence, our expectation is going to change as seen in eq:6.4.7 to the following,

$$E[g(S(t_1), S(t_2), ..., S(t_n))] = \tilde{E}[g(S(t_1), S(t_2), ..., S(t_n)) \prod_{j=1}^{n} \frac{f(Y_i)}{h(Y_i)}] \quad \text{(eq:6.4.8)}$$

where, as before, our expectation E is the expectation under the original measure and $\tilde{E}$ denotes the expectation under the new measure and the

expectation under the original measure will be finite. From eq:6.4.8 we are able to deduce that the stock prices $S(t_1), S(t_2), ..., S(t_n)$ are reliant on the independent and identically distributed random vectors $Y_1, Y_1, ..., Y_n$.

Importance sampling for path dependent options

Previously, we discussed employing importance sampling for general or vanilla options and how it relates to path simulation. However, we now consider to reduce the variance of path dependent options. We assume that the models of the underlying assets are driven by a Brownian motion. The drift parameter of the Brownian motion can therefore be changed to drive the underlying assets into important regions, with "importance" determined by the payoff of the path dependent option [36].

This method is thouroghly explained in Glasserman, Heidelberger and Shahabuddin (1999) [28]. Because this is not part of the aim of this dissertation, we will only briefly discuss the following method.

This method restricts the drift of the Brownian motion to deterministic changes over discrete time steps.

When we consider the following simulations we restrict ourselves to a discrete time grid $0 = t_0 < t_1 < t_2 < ... < t_{n-1} < t_n = T$ and we assume that our source of randomness will be a d-dimensional Brownian motion. We denote our independent standard normal random vectors as $Z_1, Z_2, ..., Z_n$. We simulate the increment of the Brownian motion by considering the time increment from time t_{i-1} to t_i as $\sqrt{t_{i-1} - t_i} Z_i$. We are now able to create Z, which consists of Z_i with length nd. "Each outcome of Z determines a path of underlying assets or state variables, and each such path determines the discounted payoff of an option. If we let H denote the composition of these mappings, then H(Z) is the discounted payoff derived from Z. Our task is to estimate E[H(Z)], the expectation taken with Z having the n-dimensional standard normal distribution." [36]

Let us then consider an example of an Asian call option, that is calculated with an arithmetic average $\bar{S}$ of all possible $S(t_i)$. We make the assumption that the underlying is modeled as a Geometric Brownian motion or $GBM(r, \sigma^2)$ and the underlying stock prices is simulated using the following;

$$S_t = S_{t_{i-1}} e^{((r-\frac{\sigma^2}{2})(t_i - t_{i-1}) + \sigma\sqrt{t_i - t_{i-1}} Z_i)} \qquad ; \forall\, i = 1, 2, 3, .., n$$

Hence, for pricing the Asian call option we view the payoff of the option as a function of Z_i,

$$H(Z) = H(Z_1, Z_2, ..., Z_n) = e^{-rT}[\bar{S} - K]^+$$

where we would price the option by finding $E[H(Z)]$ and furthermore $Z \sim N(0, I)$.

6.5 Quasi-Monte Carlo

Quasi-random sequences, also known as a low-discrepancy sequences, have a property that for all values of k, that its subsequence $x_1, x_2, \ldots, x_k$ has a low discrepancy. In layman's terms, the discrepancy of a certain sequence that belong to an arbitrary set, say A, is low if there are a certain number of points, belonging to the sequence, which is proportional to the measure of A. In section 8.5.1 we will discuss the definition of Discrepancy more rigorously. [49] Quasi-random sequences are a very common replacement for uniformly distributed random numbers. The quasi modifier is used to denote that these specific low discrepancy sequences are neither random nor pseudo-random. But, indeed these Quasi-random sequences have some properties from random variables and in certain cases of applications, such as in our case with the Quasi-random Monte Carlo method, their low discrepancy provides an advantage.

[49]"Sub random numbers have an advantage over pure random numbers

in that they cover the domain of interest quickly and evenly. They have an advantage over purely deterministic methods in that deterministic methods only give high accuracy when the number of data points is pre-set whereas in using sub random sequences the accuracy typically improves continually as more data points are added, with full reuse of the existing points."
Monte Carlo simulation is the application of a transformation to a input sequence of independent Uniformly distributed random variables $U_1, U_2, U_3, ...$ We make the assumption that d is the upper bound for the number of simulations that is needed to produce an output and we denote that the function $f(U_1, U_2, U_3, .., U_d)$ will produce this output. If we investigate where our problem is to price an option, f will be the transformation that converts the independent Uniform random variables to normal random variables and the normal random variables to the underlying asset paths and thereafter to the discounted payoff of the respective option. As we have defined before, our objective here is to define and calculate

$$E[f(U_1, U_2, U_3, .., U_d)] = \int_{[0,1)^d} f(x)dx$$

Just as Standard Monte Carlo approximates the function f, Quasi-Monte Carlo approximates the function, f, as an average of the over all points $x_1, x_2, \ldots, x_n$ and is denoted as,

$$\begin{aligned} E[f(U_1, U_2, U_3, .., U_d)] &= \int_{[0,1)^d} f(x)dx \\ &\approx \frac{1}{n}\sum_{j=1}^{n} f(x_i) \end{aligned} \qquad \text{(eq:6.5.1)}$$

where the sequence of points $x_1, x_2, \ldots, x_n$ is in the unit hypercube $[0, 1)^d$. There are some relevant comments that need to be discussed regarding the Quasi-Monte Cartlo estimate, according to Glasserman[2013] [36]

1. The approximation of the objective function, f, does not need to be in an explicit form. Monte Carlo and Quasi-Monte Carlo works in such a way that we only need an algorithm in the approximation of the function of f. This approximation algorithm is better known as the simulation algorithm.

2. In standard Monte Carlo, it will have no influence on the approximation of f whether we include the boundary of the unit hypercube or not. The value of the integral under standard Monte Carlo will remain unchanged as denoted in eq:6.5.1. Quasi-Monte Carlo on the other hand requires the specification of the set of points to which a boundary belongs too. It is, however, standard practice and convenient to evaluate the integrals to be open on the right of the hypercube and to be closed on the left of the hypercube. Hence, our use of the unit hypercube and is denoted as $[0,1)^d$.

3. In standard Monte Carlo simulation, if we simulated Uniform independent and identically distributed random variables $U_1, U_2, U_3, \ldots$ and we created the following vectors $(U_1, U_2, U_3, \ldots, U_d)$, $(U_{d+1}, U_{d+2}, U_{d+3}, \ldots, U_{2d})$, $(U_{2d+1}, U_{2d+2}, U_{2d+3}, \ldots, U_{3d})$, ... These vectors now produce an independent and identically distributed sequence of points from a $d-$dimensional hypercube. In Quasi-Monte Carlo, the dimension of the problem will indicate how the construction of points x_i will be done. Hence, the vectors x_i in $[0,1)^d$ cannot be constructed by taking sets of d consecutive elements from a scalar sequence.

Quasi-Monte Carlo [36] is dependent on the dimension of a problem and the dimension of the problem should be known before we are able to evaluate the problem. This dependence on dimension is one of the most discerning characteristics of Quasi-Monte Carlo simulation. If we consider, for example, two different Monte Carlo algorithms with corresponding approximation functions, lets say f and g. Where f is defined as a function $f : [0,1)^{d_1} \to \mathbb{R}$

and g is defined as $g : [0,1)^{d_2} \to \mathbb{R}$ which results in $f(U_1, U_2, ..., U_{d_1})$ and $g(U_1, U_2, ..., U_{d_2})$. This will result that both approximations will have the same distribution, with the same variance properties and bias. The ideal Monte Carlo algorithm would be one that requires less time to calculate the approximations, and the dimensions d_1 and d_2 should be irrelevant to the calculation except influencing computational time of the approximations. When using standard Monte Carlo, the dimension of the problem does not need to be specified, but with Quasi-Monte Carlo one needs to determine the dimension of a problem before one can start the generation of Quasi random numbers. Quasi-Monte Carlo usually performs better than standard Monte Carlo under lower dimensional representations.

6.5.1 Discrepancy

If we want to fill a hypercube uniformly, it would make sense that we attempt to choose a certain point x_i to lie on a grid. Grids, however, have severe shortcomings. If we have that our function f is a nearly separable function of its d arguments, then the information of the values of f at n^d is the almost the same as the information contained of the f values at nd. Major drawbacks of a grid is that it will leave large rectangles without containing any point over the unit hypercube $[0,1)^d$ and the number of points needs to be specified in advance. If the aim is to refine the grid over the unit hypercube by adding additional points, the points, which must be added, to be able to find the next favorable approximation grow very quickly.

For example [36], if we construct a grid as a Cartesian product of 2^k point along each of the d dimensions. We then obtain a total of 2^{kd} points. Now we attempt to fill each gap in each of the d dimensions with an extra point, which means doubling the number of points per dimension. Hence, the total number of new points added to the grid is $2^{(k+1)d} - 2^{kd}$ and grows very quickly with k.

We now define the notation for the deviation from uniformity, which we

measure through various notations of discrepancy. Given a collection $\mathbf{A}$ of (Lebesgue measurable) [36] subsets of $[0,1)^d$ the discrepancy of the point set $\{ x_1, x_2, ..., x_n \}$ relative to $\mathbf{A}$ is defined by,

$$D(x_1, x_2, ..., x_n; \mathbf{A}) = \sup_{A \in \mathbf{A}} |\frac{\#\{ x_i \in A \}}{n} - vol(A)|$$

where $\#\{ x_i \in A \}$ denotes the number of points x_i that is contained within A and $vol(A)$ is defined as the volume of A. Thus, discrepancy by this expression can be interpreted as the supremum over the errors by integrating the indicator function of A using the points $x_1, x_2, ..., x_n$.
If we now define $\mathbf{A}$ to be the collection of all rectangles over the unit hypercube interval $[0,1)^d$ of the form,

$$\prod_{i=1}^{d} [u_i, v_i) \quad , 0 \leq u_i \leq v_i \leq 1,$$

yields the standard discrepancy(or extreme) $D(x_1, x_2, ..., x_n)$. If we only restrict $\mathbf{A}$ to rectangles then,

$$\prod_{i=1}^{d} [0, u_i)$$

will define the star discrepancy $D^*(x_1, x_2, ..., x_n)$.
Niederreiter [30] in 1992 proved that the ordinary or standard discrepancy has an upper and lower limit with relation to the definition with the star discrepancy. Niederreiter in proposition 2.4 proved that,

$$D^*(x_1, x_2, ..., x_n) \leq D(x_1, x_2, ..., x_n) \leq 2^d D^*(x_1, x_2, ..., x_n) \qquad \text{(eq:6.5.2)}$$

where we have that for a fixed d the quantities will have the same order of magnitude.
If we require true uniformity then it is logical that we need these discrepancy

measures to be as small as possible. However, both of these D and D^* measures focus on the products of intervals and will typically ignore a rotated subcube of the unit hypercube. If we still work under the assumption that the integrand f is the simulation algorithm, the coordinate axis may not yield meaningful results. It would therefore be meaningful to look at points that achieve low results in both these measures of discrepancy; that is the main goal of what low discrepancy methods does.

The next results are from Niederreiter [30]. If we work in the first dimension and set $d = 1$. Niederreiter showed that the following holds.

$$D(x_1, x_2, ..., x_n) \geq \frac{1}{n}$$

and

$$D^*(x_1, x_2, ..., x_n) \geq \frac{1}{2n}$$

in both cases the minimum is obtained and the equality holds for,

$$x_j = \frac{2j-1}{2n} \qquad , j = 1, 2, 3, \ldots, n \tag{eq:6.5.3}$$

For this specified set of points, the eq:6.5.1 will be reduced to the midpoint rule of integration over the unit interval. We notice that in eq:6.5.3 the first n defined points have no values in common with the following set of points for $n + 1$.

If we now construct a sequence of infinite points, say $x_1, x_2, x_3, \ldots$ defined in the unit interval $[0, 1)$ and we aim to measure the discrepancy of the first k defined points of this infinite sequence. If we consider the numerical integration as defined in eq:6.5.1 the accuracy of this integration approximation will be dependent on the amount of points that we define on the unit interval $[0, 1)$.

Niederreiter [30] cites that Schmidt in 1972 showed that the following holds,

$$D(x_1, x_2, ..., x_n) \geq D^*(x_1, x_2, ..., x_n) \geq \frac{c \log n}{n}$$

where the sequence $x_1, x_2, ..., x_n$ is defined over $[0, 1)$ and for an absolute constant $c > 0$. Thus, for low discrepancy sequences in the one-dimensional case we cannot expect something better than,

$$D^*(x_1, x_2, ..., x_n) = O(\frac{\log n}{n}) \qquad , \forall\, n \geq 2$$

By fixing the number of points in advance, in other words using the first finite n number of points of a sequence rather than generating a different set of points. Hence, this will have the implication of increasing the discrepancy of the sequence by a factor $\log n$.

If we now consider multiple dimensions, there is less known about finding the best possible discrepancy in higher dimensions, $d > 1$. Niederreiter [30] states that it is widely believed that for an $s-$dimensional case and $s \geq 2$, the star discrepancy of any $N-$element point set satisfies

$$D^*(x_1, x_2, ..., x_N) \geq \mathbf{a_s} \frac{(\log N)^{s-1}}{N}$$

where $\mathbf{a_s} > 0$ is a constant that is dependent only on the dimension s. For any infinite set of points $x_1, x_2, x_3, ...$, the first N elements of the sequence $x_1, x_2, ..., x_N$ satisfies

$$D^*(x_1, x_2, ..., x_N) \geq \mathbf{a'_s} \frac{(\log N)^{s}}{N}$$

where $\mathbf{a'_s} > 0$ is a constant that is dependent only on the dimension s.

These order-of magnitude discrepancies are achieved by explicit construction in the following section. Where the discussion continues for Halton, Faure and Sobol sequences. Quasi sequences that achieve a star discrepancy of $O((\frac{\log n)^s}{n})$ is therefore called low discrepancy sequences.

If we obtain a large finite sequence with n points, the power of $\log n$ becomes negligible with respect to n, but this is only true for when the dimension of

the problem is relatively small. Quasi-Monte Carlo has been traditionally characterized as appropriate for problems in higher dimensions.

6.5.2 Pseudo-random sequences

A Pseudo random number generator is a deterministic algorithm that is used to create a sequence of numbers with little or no discernable pattern in the generated numbers, except for broad statistical properties [25]. A Pseudo random number generator is also known as a Deterministic Random Bit generator or DRBG.
According to Niederreiter [30] - "The success of a Monte Carlo calculation depends, of course, on the appropriateness of the underlying stochastic model, but also, to a large extent, on how well the random numbers used in the computation simulate the random variables in the model."
Desirable properties that we wish Pseudo random sequences to display are as follows,

1. the smallest period length should be sufficiently large

2. it should have little intrinsic structure(for example lattice structure)

3. the Pseudo random sequence should have good statistical properties

4. the Pseudo random sequence should be computationally efficient to calculate

We also have to make the distinction between uniform Pseudo random number and non-uniform Pseudo random numbers. The uniform pseudo random number generator will simulate uniform distributed random numbers over the unit interval $[0, 1)$. Non-uniform Pseudo random numbers are usually simulated from a uniform Pseudo random number and using a sampling method the distribution of the uniform distribution is changed to the target or desired distribution.

Linear Congruential Pseudo random numbers

As we have discussed in section 6.2.1. The Linear Congruential generator is a method for producing Pseudo random numbers. The mixed Linear Congruential generator (LCG) was first proposed by Lehmer [1951] [29]. The parameters for the Linear Congruential generator is as follows. We choose a large positive integer m, an integer k for $k : 1 \leq k < m$ and finally $c \in \{0, 1, \ldots, m-1\}$. We then select a seed value or initial value that is denoted by $a_0 \in \{0, 1, \ldots, m-1\}$. We are then able to generate the values $a_1, a_2, a_3, ...$ with the following recursion formula,

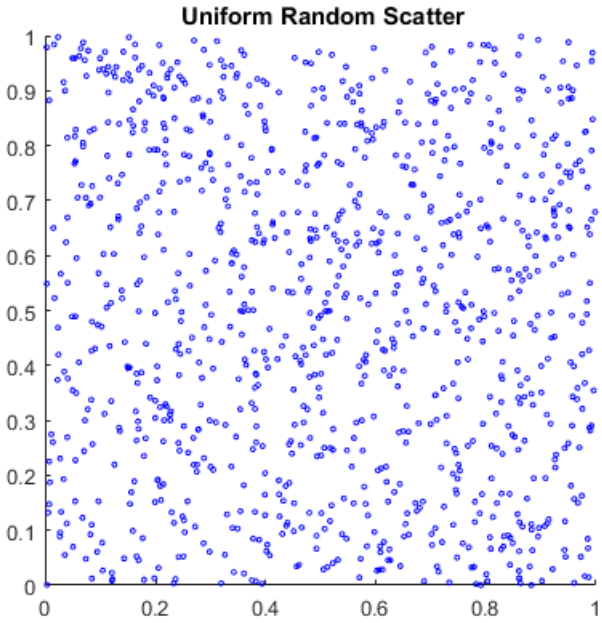
$$a_{i+1} = (ka_i + c) mod(m) \qquad , i = 0, 1, 2, \ldots$$

From these generated values $a_1, a_2, a_3, ...,$ we are able to find the Linear Congruential Pseudo random numbers,

$$u_{i+1} = \frac{a_{i+1}}{m} \in [0, 1) \qquad , i = 0, 1, 2, \ldots$$

Example and graph Please refer to section 7.2.1 for the mixed Linear Congruential example for generating Pseudo random numbers. We now plot a 1000 points of the Linear Congruential Pseudo random points in order to make a relavant comparison between these Pseudo generated numbers and Quasi generated random numbers.

Figure 6.4: Plot of a computer generated random numbers sequence with 1000 points



We observe that these computer generated uniform random variables are not truly evenly distributed over the unit interval. There occurs some clumping of points over the interval and we expect that this clumping of points will influence both accuracy and the rate of convergence. We expect greater results, in this regards, towards our plotting of the Quasi- Monte Carlo sequences or low discrepancy sequences.

6.5.3 Van der Corput sequence

The first Quasi random sequence that we discuss is the Van der Corput sequence. The reason for this is that other Quasi sequences, such as the Faure and Halton sequences, are multi dimensional expansions of the Van der

Corput sequence. Hence, we introduce this one-dimensional low discrepancy sequence called the Van der Corput sequence. Firstly, we define a $base - b$ as an integer value that is greater or equal to two. So, for every integer k^+ will have a unique representation as a linear combination of non-negative powers of b with coefficients in $\{0, 1, 2, 3, \ldots, b-1\}$. We are able to obtain,

$$k = \sum_{i=0}^{n} c_i(k) b^i$$

where we get finite number of coefficients $c_i(k)$ that are not equal to zero. The function ϕ is then defined as the function that maps every k to a point in $[0, 1)$. The function is defined as follows,

$$\phi_b(k) = \sum_{h=0}^{\infty} \frac{a_h(k)}{b^{h+1}}$$

Example If we calculate a Van der Corput sequence with $base = 2$ and $k = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Then we calculate the first 8 entries of the Van der Corput sequence. Then we obtain a k binary sequence with respect to every defined k,

$$k \; in \; binary = \{0, 1, 10, 11, 100, 101, 110, 111\}$$

and then,

$$\phi_2(k) \; binary \; = \{0, 0.1, 0.01, 0.11, 0.001, 0.101, 0.011, 0.111\}$$

and finally we get Van der Corput sequence given as,

$$\phi_2(k) \; = \{0, 1/2, 1/4, 3/4, 1/8, 5/8, 3/8, 7/8\}$$

6.5.4 Halton sequence

In a one-dimensional setting the Halton sequence becomes the Van der Corput sequence. Firstly, we discuss the Van der Corput sequence and then look at the multi-dimensional case, which becomes the Halton sequence.
The Van der Corput sequence can be seen as the following equation:

$$n = \sum_{k=0}^{L-1} d_k(n)\, b^k; n \geq 1$$

where b is the base number of which n is represented, and $0 \leq d_k(n) \leq b$. This then maps onto the unit interval $[0, 1)$. The corresponding rate of convergence is $O\left(\frac{\log(n)}{n}\right)$.
Halton sequences are mostly used to generate points in a specific space for the use of numerical methods, for example, the Monte Carlo simulation. These sequences are deterministic in nature, but they are of low discrepancy. This means that the sequences appear to be random and can be applied in different situations. To generate the Halton sequence we generalise the Van der Corput sequence to a multi-dimensional case. Hence, for the multi-dimensional case we obtain the equation:

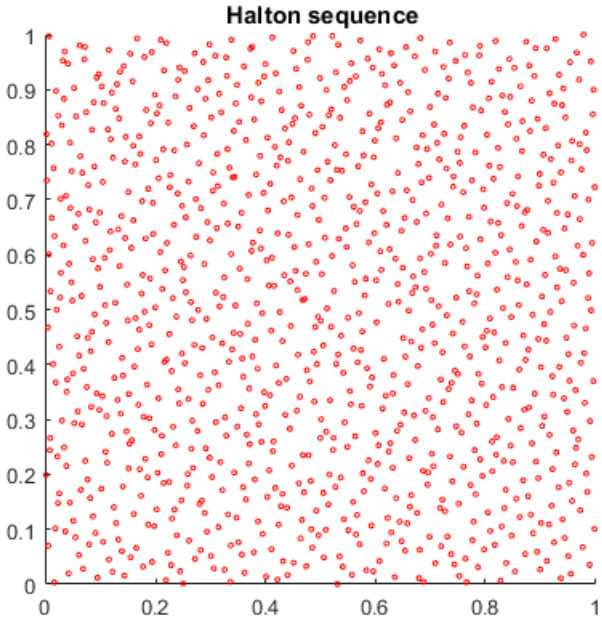
$$n = \sum_{k=0}^{L-1} \overline{d}_k(n)\, b^k; n \geq 1$$

where $\overline{d}_k(n)$ now represents a vector, in the form of, $\overline{d}_k(n) = (\Phi_{b1}(n-1), \ldots, \Phi_{bt}(\mathrm{n}-1))$. The star discrepancy of this sequence is $E_n^* = O(\frac{(\log(n))^t}{n})$
Thus, we follow with a scatter plot of a Halton sequence, to observe the random nature of the sequence and the reason for being effective randomization tool. In Figure 6 below, we constructed a scatter plot of all the Halton Quasi-random sequence points. We can deduce from this plot that there are no clustering of points and all point plotted seems to be uniformly distributed over the hypercube of [0,1). In a more statistical setting, if we

test for randomness, most of the Quasi-random sequences are too uniform to pass these tests [26].

Halton sequence graph

Figure 6.5: Plot of a Halton sequence with 1000 points



If we compare the graphs of figure 7 and figure 4, we deduce that from figure 4 there are points present that cluster together, which decrease the probability of true randomness of these points. But in the Halton Quasi-random number sequence we see a more evenly spread of points over the unit interval, which leads to the conclusion of more truly random points.

6.5.5 Faure sequence

In 1982, Faure [42] proposed to develop a different extension of the Van der Corput sequence to multiple dimensions. A common base is to be used across all simulated dimensions. There is, however, a restriction on the value of the chosen base, the base must be at least as large as the dimension itself .
If we now specify a $d-$dimensional Faure sequence, according to Glasserman [36], then the coordinates of the Faure sequence are obtained by permuting segments of a single Van der Corput sequence. In the choice for the base b we choose the smallest prime number greater or equivalent to the magnitude of d. We let $c_i(k)$ denote the coefficients in the base b expansion of k, so that it is given in the following expression,

$$k = \sum_{i=0}^{\infty} c_i(k) b^i \qquad \text{(eq:6.5.6)}$$

Hence, we obtain the j^{th} coordinate $j = 1, 2, ..., d$ of the k^{th} point in the Faure sequence is expressed as,

$$\sum_{h=1}^{\infty} \frac{y_h^{(j)}(k)}{b^h} \qquad \text{(eq:6.5.7)}$$

where $y_h^{(j)}$ is defined as

$$y_h^{(j)}(k) = \sum_{i=0}^{\infty} \binom{i}{h-1} (j-1)^{i-h+1} a_i(k) mod(b) \qquad \text{(eq:6.5.8)}$$

where $\binom{i}{h-1}$ denotes a permutation function.
Hence, each of these defined sums in eq:6.5.6, eq:6.5.7, and eq:6.5.8 has only a finite number of nonzero elements. If we specify the base-b expansion of k with r number of terms, resulting that $a_{r-1}(k) \neq 0$ and $a_i(k) = 0 \, \forall \, i \geq r$. Hence, we are able to view eq:6.5.8 as a matrix- vector representation and we are able to calculate the sequence,

$$\begin{pmatrix} y_1^{(j)} \\ y_2^{(j)} \\ y_3^{(j)} \\ \dots \\ y_r^{(j)} \end{pmatrix} = C^{(j-1)} \begin{pmatrix} a_0(k) \\ a_1(k) \\ a_2(k) \\ \dots \\ a_{r-1}(k) \end{pmatrix} mod(b)$$

and thus, from eq:7.5.8 the matrix C^j will be a matrix that calculates the permutations and is an rxr matrix. Defined as,

$$C^j(q,s) = \binom{s-1}{q-1} j^{s-q} \qquad , for\ s \geq q\ and\ 0\ otherwise$$

where (q, s) specifies the entries for the matrix C. There is also a special relationship that holds for the matrix C. This will increase the efficiency of the calculation,

$$C^j = C^1 C^{j-1}$$

Pseudo code for Faure sequence [4]with dimension d and base b

1. We set $q = (\frac{\ln(N)}{\ln(b)})$. We then define $b = (b^{-1}, b^{-2}, ..., b^{-q})$ and we set $n = 0$.

2. We then calculate the b representation $(a_q...a_1)$ of n. Then create a vector for $a = (a_1, a_2, ..., a_q)^T$

3. Then for $i = 1, 2, ..., d$ calculate $a_k = G^{k-1}a$

4. Also for $i = 1, 2, ..., d$, we calculate $x_{nk} = ba_k$ and let $x_n = (x_{n1}, ..., x_{nd})^T$ be the n'th Faure point

5. So then if $n = N$ then we stop. Otherwise, set $n = n + 1$ and return to step 2

```
1 %The code from DP. Kroese, T Taimre and ZI Botev[4]
2  was used to generate the Faure sequence
```

```
 3
 4 function S = faure1(q,d,M)
 5 z = floor(log(M)/log(q))+1;
 6 a=repmat((0:M)',1,z);
 7 S=zeros(M+1,d);
 8 F=zeros(z,z);
 9 b=repmat(1./q.^(1:z),M+1,1);
10 S(:,1)=sum(b.*a,2);
11 for j=1:z
12 for i=1:j
13 G(i,j)=mod(nchoosek(j-1,i-1),q);
14 end
15 end
16 G=G';
17
18 for i=1:z-1
19 a(:,i)=mod(a(:,i),q);
20 a(:,(i+1):z)=floor(a(:,(i+1):z)/q);
21 end
22 for k=2:d
23 a=mod(a*G,q) ;
24 S(:,k)=sum(b.*a,2) ;
25 end
```

Example and graph

We follow and rework the example from Glasserman [36], Consider the case where $r = 2$ and $b = 3$. Then we are able to find the corresponding C matrices,

$$C^{(1)} = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right),\ C^{(2)} = C^{(1)}C^{(1)} = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right)$$

Our corresponding $\bar{a}(k)$ vectors for $k = 0, 1, 2, 3, 4, 5, 6, 7, 8$ are

$$\binom{0}{0}, \binom{1}{0}, \binom{2}{0}, \binom{0}{1}, \binom{1}{1}, \binom{2}{1}, \binom{0}{2}, \binom{1}{2}, \binom{2}{2}$$

We are now able to calculate the corresponding vectors for $C^{(1)}a(k)mod(b)$ and $C^{(2)}a(k)mod(b)$ respectively,

$$C^{(1)}\bar{a}(k)mod(b) = \binom{0}{0}, \binom{1}{0}, \binom{2}{0}, \binom{1}{1}, \binom{2}{1}, \binom{0}{1}, \binom{2}{2}, \binom{0}{2}, \binom{1}{2}$$

$$C^{(2)}\bar{a}(k)mod(b) = \binom{0}{0}, \binom{1}{0}, \binom{2}{0}, \binom{2}{1}, \binom{0}{1}, \binom{1}{1}, \binom{1}{2}, \binom{2}{2}, \binom{0}{2}$$

Now, using eq:7.5.7 to obtain a vector $(1/3, 1/9)$. We then multiply this vector by each of the above three sequence of vectors. Hence, we obtain,

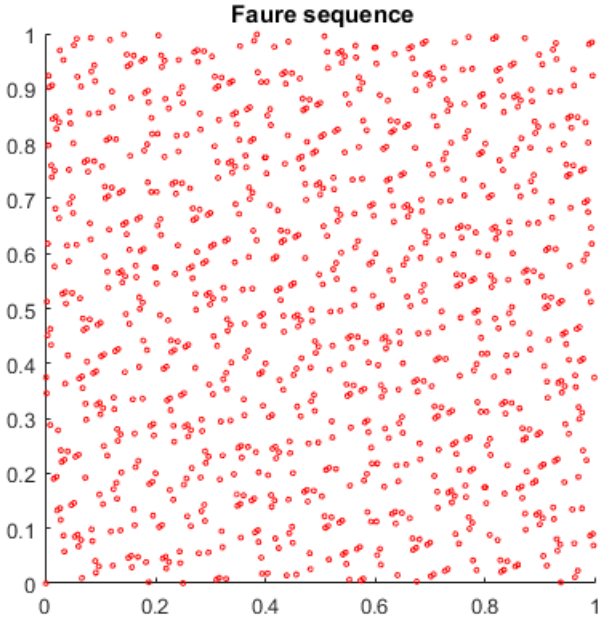
$$\{0, 1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9\}$$

$$\{0, 1/3, 2/3, 4/9, 7/9, 1/9, 8/9, 2/9, 5/9\}$$

$$\{0, 1/3, 2/3, 7/9, 1/9, 4/9, 5/9, 8/9, 2/9\}$$

The first row yields the first 9 elements from the Van der Corput sequence with a specified base 3. If we view these points now in a three-dimensional space the columns of the sequences will yield the respective coordinates to be plotted. Hence, the first four coordinates will be $[0, 0, 0]$, $[1/3, 1/3, 1/3]$,$[2/3, 2/3, 2/3]$ and $[1/9, 4/9, 7/9]$. Thus, we are able to obtain the first nine points of the three-dimensional Faure sequence.

Figure 6.6: Plot of a Faure sequence with 1000 points



6.5.6 Sobol sequence

Sobol sequences are once again an example of Quasi-random low discrepancy sequences. This specific quasi-random sequence was first introduced by the Russian mathematician Sobol in 1967. These Sobol sequences make use of the base of two, to form finer uniform partitions of the unit interval or, in a multi-dimensional sense, the unit hypercube. But the coordinates are finally reordered in each dimension [50].

Let the following be an n-dimensional hyper cube, $[0,1)^n$, with the function g that is integrable over the unit hypercube $[0,1)^n$. Hence, the primary idea

of Sobol was to construct a sequence y_n in the hypercube $[0,1]^n$, so that:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} g(y_j) = \int_{[0,1]^n} g$$

For the generation of the Sobol sequence we follow Joe and Kuo. [52] To generate the k'th component of the point in a specific Sobol sequence, it will require us to choose a polynomial with degree t_k in the field of $\mathbb{Z}_2$, hence we have that;

$$y^{\,t_k} + c_{1,k} y^{\,t_k-1} + c_{2,k} y^{\,t_k-2} + \cdots + c_{\,t_k-1,k} x + 1$$

where $c_{1,k}, \ldots, c_{\,t_k-1,k}$ are the coefficients of the polynomial and are either 0 or 1. We will then use these coefficients $c_{1,k}, \ldots, c_{\,t_k-1,k}$ to create a sequence $\{n_{1,k},\ n_{2,k}, \ldots\}$, with the recurrence relation that will consist only of positive integers.

$$n_{p,k} = 2c_{1,k} n_{p-1,k} \oplus 2^2 c_{2,k} n_{p-2,k} \oplus \cdots \oplus 2^{\,t_k-1} c_{\,t_k-1,k} n_{p-\,t_k+1,k} \oplus 2^{\,t_k} n_{p-\,t_k,k} \oplus n_{p-\,t_k,k}$$

For $p \geq\ t_k{+}1$, where $\oplus$ is a bit by bit operator. The values of $n_{1,k},\ n_{2,k}, \ldots, n_{\,t_k,k}$ can be chosen randomly but need to uphold the following constraints $n_{v\ ,k},\ 1 \leq v \leq\ t_k$, is odd, and less than 2^v. The "direction number" is then defined as follows

$$w_{v,k} := \frac{n_{v,k}}{2^v}$$

Then finally, we obtain the $y_{i,k}$, which is the k'th component of the i'th point of the Sobol sequence, and is given by;

$$y_{i,k} = \ b_1 w_{1,k} \oplus b_2 w_{2,k} \oplus \ldots$$

where b_l is the l'th bit from the right when i is written in binary, that is, $(\ldots b_2 b_1)_2$. There is also a more efficient way of generating a Sobol sequence.

It is called the Gray code, introduced by Antonov and Saleev [1979].

Example and graph

Consider the following example from Glasserman [36] to better illustrate the process of generating a Sobol sequence.
Let us consider the primitive polynomial,

$$x^3 + x^2 + 1$$

with degree $t_k = 3$. The recurrence algorithm becomes

$$n_{p,k} = 2n_{p-1,k} \oplus 8n_{p-3,k} \oplus n_{p-3,k}$$

and assume that we initialize the recurrence with $n_{1,k} = 1, n_{3,k} = 3, n_{3,k} = 3$. Then the next two elements will be calculated as follows,

$$\begin{aligned} n_{4,k} &= (2 \cdot 3) \oplus (8 \cdot 1) \oplus 1 \\ &= 0110 \oplus 1000 \oplus 0001 \\ &= 1111 \\ &= 15 \end{aligned}$$

$$\begin{aligned} n_{5,k} &= (2 \cdot 15) \oplus (8 \cdot 3) \oplus 3 \\ &= 11110 \oplus 11000 \oplus 00011 \\ &= 00101 \\ &= 5 \end{aligned}$$

From these 5 values for $n_{p,k}$, we are now able to calculate the corresponding values for $w_{p,k}$ by dividing $n_{p,k}$ by 2^p. But there exists an equivalence that instead of dividing by 2^p we shift the binary point to the left by p places in the representation of $n_{p,k}$. Thus, we are able to obtain the first 5 direction

numbers,

$$
\begin{aligned}
w_{1,k} &= 0.1 \\
w_{2,k} &= 0.11 \\
w_{3,k} &= 0.011 \\
w_{4,k} &= 0.1111 \\
w_{5,k} &= 0.00101
\end{aligned}
$$

and we are able to find the corresponding generator matrix,

$$
V = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$

thus we now calculate the sequence $x_1, x_2, ...$ For each k we are able to take a vector $a(k)$ of binary coefficients of k and pre-multiply it by the matrix V. The resulting vectors give the coefficients of a binary fraction. Hence, we obtain the first three vectors,

$$
V \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, V \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, V \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

which yield the first three points of $1/2, 3/4$, and $1/4$ and the last three points

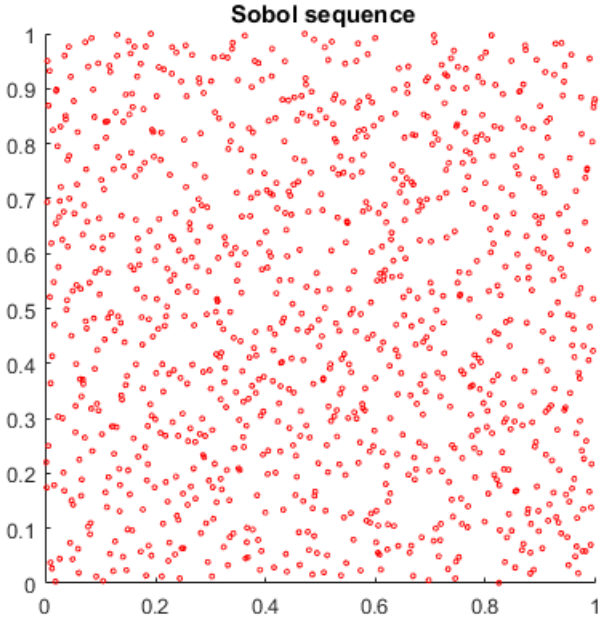
that can be generated from the matrix V are

$$V\begin{pmatrix}1\\0\\1\\1\\1\end{pmatrix}=\begin{pmatrix}0\\0\\1\\1\\1\end{pmatrix},V\begin{pmatrix}0\\1\\1\\1\\1\end{pmatrix}=\begin{pmatrix}0\\1\\1\\1\\1\end{pmatrix},V\begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix}=\begin{pmatrix}1\\1\\1\\1\\1\end{pmatrix}$$

which produces the last three points as $7/32, 15/32$, and $31/32$.

In figure 7.7 below we plot 1000 points. As one is able to observe there are minimal clumping of points and the points seem to be evenly distributed over the unit interval. As we compare this figure 5 to the figure of Pseudo random numbers, we are able to deduce that there are more clumping of points under Pseudo random sequences relative to the Sobol sequence. We expect that these remarks will have some influence on accuracy and on the rate of convergence.

Figure 6.7: Plot of a Sobol sequence with 1000 points



For the code to generate the Sobol sequence, refer to the appandix or D.P. Kroese, T. Taimre and Z.I. Botev [4].

6.6 Example

The following example will compare European Black-Scholes estimate with Quasi-Monte carlo estimates. We see in Table 7.2 that all sequences got good approximation except the Halton sequence. As deduced in the Glasserman [36] paper we expected that the Halton sequence will perform the worst of all the Quasi-sequences and the the Sobol sequence will produce unconditionally stable estimates.

Table 6.2: European call estimates for Quasi-random sequences

Strike K	BS est.	CG est.	Sobol est.	Halton est.	Faure est.
80	37.730	37.737	37.746	40.468	38.03081
85	33.358	33.360	33.370	36.050	33.64382
90	29.113	29.117	29.124	31.738	29.28317
95	25.056	25.067	25.062	27.584	24.97179
100	21.249	21.265	21.248	23.636	20.75152
105	17.748	17.764	17.738	19.958	16.66818
110	14.597	14.598	14.578	16.603	12.80963
115	11.823	11.822	11.803	13.607	9.427844
120	9.434	9.451	9.423	10.983	6.876027
125	7.419	7.455	7.410	8.716	5.221166
130	5.754	5.792	5.752	6.831	4.245497
135	4.405	4.439	4.409	5.298	3.612412
140	3.331	3.353	3.333	4.068	3.152193

avg relative error	0.016	0.009	1.887	1.064
avg % rel error	0.093%	0.055%	11.098%	5.635%
Rate of convergence	0.023	0.043	0.889	2.765

[The above results are obtained for European call options for different strike prices. We used Sobol-, Halton- and Faure-sequences to obtain these estimates. As well as CG(computer generated) random samples for estimation. In this example our parameters were as follows; r=0.1, $\sigma = 0.2$, $S_0 = 110$ and time to maturity was 1 year. We ran 10 000 simulations on 100 paths and to obtain rate of convergence, we also ran 1 000 simulations on 100 paths.]

The following figures 6.8, 6.9 and 6.10 are plots that represent the Quasi estimates, as they converge to the Black-Scholes estimate, for European

options. Quasi-Monte Carlo sequences enables us to improve on the reliability and rate of convergence of Monte Carlo simulation. Quasi-random sequences are deterministic sequences by nature, which results in deterministic error bounds and improved convergence. In this case, we studied the Halton, Faure and Sobol sequences. According the Glasserman [36], both the Halton and Faure sequences deliver marginal better results, compared to normal Monte Carlo. However, the Sobol sequence delivers significant better convergence and reliability to Monte Carlo simulation. This is due to the Sobol sequence being more evenly dispersed over the unit hypercube, compared to the other sequences.

Figure 6.8: Plot of Quasi-sequences and CG generated estimates

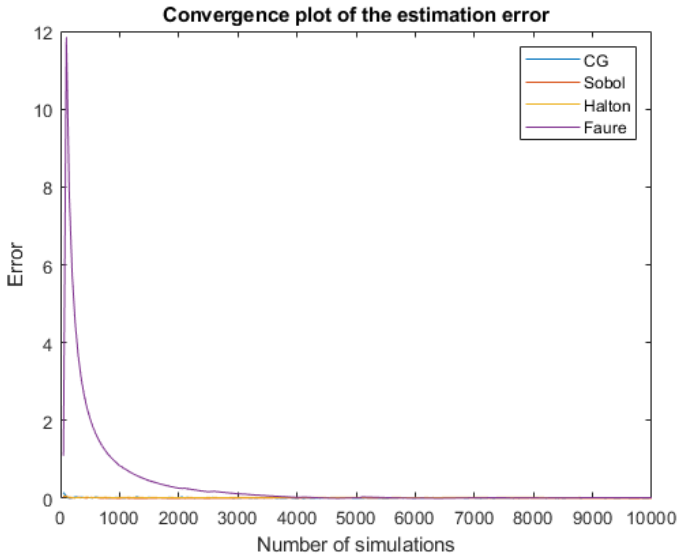


Figure 6.9: Plot of Quasi-sequences(excluding Faure sequence) and CG generated estimates

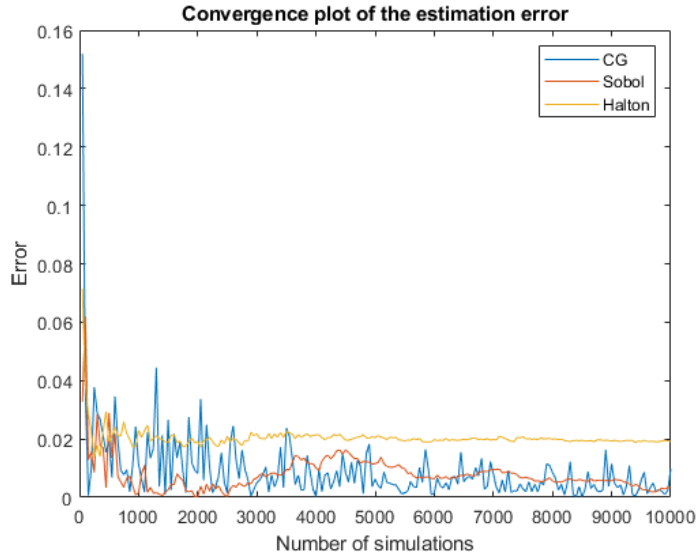
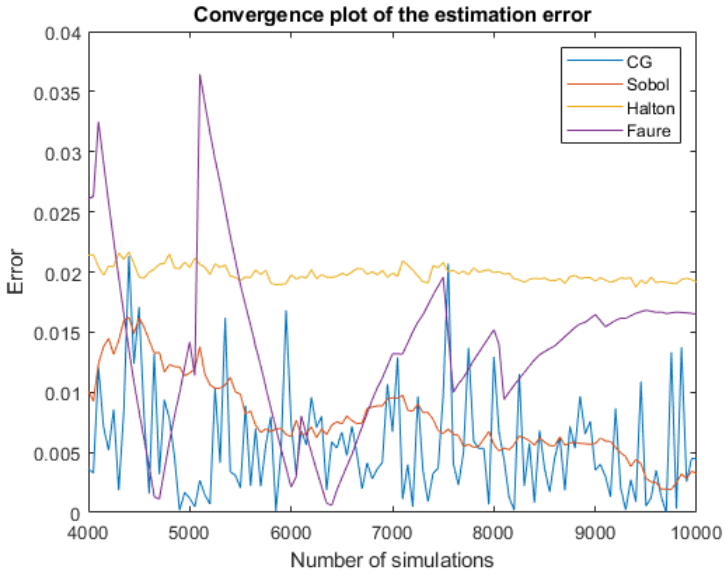


Figure 6.10: Plot of Quasi-sequences and CG generated estimates



We generated 50 to 10 000 random numbers, by an increment of 50 simulations. It can be seen from figure 6.8, that the Faure sequence has a large estimation error, for a small number of simulations. However, the Faure sequence tend to converge faster to 3000 simulations on wards. In figure 6.9 it is observed that the Sobol sequence has a small deviation and converges smoothly. The Sobol sequence delivered estimates with small relative errors, under a low number of simulations and tends to be the more stable sequence. However, the computer generated random numbers and Faure sequence tends to have a larger standard deviation.

Chapter 7

Longstaff and Schwartz

Longstaff and Schwartz [39] proposed a model in 2001 to evaluate American options by simulation. The key difference in this approach is the use of least squares to estimate the conditional payoff to the option holder from continuation. This approach is very applicable to path dependent options. This method is called least squares Monte Carlo (LSM) algorithm.
The best way to describe the technique and its workings intuitively is, firstly, to give a numerical example of the technique. The following example follows directly from the Longstaff and Schwartz(2001) paper to describe the technique. Afterwards, we will describe the algorithm of the model in a more formal manner.
To describe the workings of an American option fully, we will consider the pricing of a vanilla American option for this numerical example. We will exercise the option when we consider it to have an optimal exercise strategy; namely when it is in the money at maturity. If we consider the same scenario prior to the date of maturity, we consider the optimal strategy to be the comparison between the immediate exercise value and the expected cash flows from continuing. We would exercise the option if the immediate exercise is greater in value. Hence, to optimally exercise an American option, is to find the conditional expected value of continuation. We approach this

problem in the manner that we look at the cross-sectional information in the simulated paths of the underlying to identify the conditional expectation function. This is, however, done by regressing the realized cash flows from continuation on a set of basis functions of the values of the relevant state variables. "The fitted value of this regression is an efficient unbiased estimate of the conditional expectation function and allows us to accurately estimate the optimal stopping rule for the option." [39]

Numerical example

The best way to convey this idea on pricing is by a numerical example. Let us consider an American put option priced on an underlying, which pays no dividends.

Strike(K)=	1.10
Risk free rate(r)=	0.06
S_0=	1
e^{-rt}=	0.9417645

For simplicity's sake the stock price has only eight possible paths, which we display in a table below. The stock price paths are generated under the assumption of risk neutrality for three discrete time steps, t. Hence the paths are as follows,

Table 7.1: Stock price paths for three discrete time steps

Path	t=0	t=1	t=2	t=3
1	1	1.09	1.08	1.34
2	1	1.16	1.26	1.54
3	1	1.22	1.07	1.03
4	1	0.93	0.97	0.92
5	1	1.11	1.56	1.52
6	1	0.76	0.77	0.9
7	1	0.92	0.84	1.01
8	1	0.88	1.22	1.34

Our aim from this stock price paths table is to obtain a stopping rule that will maximize the value of the option on every discrete time step and relative path. For completeness, it is necessary to calculate all the intermediate matrices at different time steps, because we apply our method recursively. If the option was not exercised before the expiration date at time three, then the cash flows (option values) among different paths will be realized by the option holder from the following optimal strategy at time three, as seen in the table below

Table 7.2: The corresponding cash flow(expiration payoff) matrix at time 3

Path	t=0	t=1	t=2	t=3
1	-	-	-	$(1.1-1.34)^+=0.00$
2	-	-	-	$(1.1-1.54)^+=0.00$
3	-	-	-	$(1.1-1.03)^+=0.07$
4	-	-	-	$(1.1-0.92)^+=0.18$
5	-	-	-	$(1.1-1.52)^+=0.00$
6	-	-	-	$(1.1-0.90)^+=0.20$
7	-	-	-	$(1.1-1.01)^+=0.09$
8	-	-	-	$(1.1-1.34)^+=0.00$

These respective cash flows, which have been calculated at time three, would be identical for both European and American option calculations.

If the option is in the money at time two, then the option holder must decide whether to continue with the option until expiry at time three or exercise the option immediately. From our stock price paths at time two, there are only five possible paths that are in the money and that we can consider when exercising the option early.
We now define X and Y so that X represents the underlying stock prices at time two for the respective paths that are in the money. We define Y to be the discounted corresponding cash flows received at time 3, only if the put is not exercised at time 2. We will only consider in the money paths to estimate the conditional expectation function, where exercise is relevant and this will significantly improve the efficacy of the algorithm. Therefore, we are now able to obtain the values for the defined X and Y at time 2,

Table 7.3: Regression vectors at time 2

Path	Y	X
1	0.00x0.94176	1.08
2	-	-
3	0.07x0.94176	1.07
4	0.18x0.94176	0.97
5	-	-
6	0.20x0.94176	0.77
7	0.09x0.94176	0.84
8	-	-

The regression of Y on a constant X and X^2 will yield the expected cash flow from continuing with the option's life conditional on the stock price paths at time 2. This regression takes place with the implementation of a weighted Laguerre polynomial as basis functions, denoted as follows,

$$
\begin{aligned}
L_0(X) &= e^{\frac{-X}{2}} \\
L_1(X) &= e^{\frac{-X}{2}}(1-X) \\
L_2(X) &= e^{\frac{-X}{2}}(1-2X+\frac{X^2}{2}) \\
&\dots \\
L_n(X) &= e^{\frac{-X}{2}}\frac{e^X}{n!}\frac{d^n}{dX^n}(X^n e^{-X})
\end{aligned}
$$

Then we obtain the function of continuation, denoted by $F(\omega; t_{k-1})$ as follows,

$$F(\omega; t_{k-1}) = \sum_{j=0}^{\infty} a_j L_j(X),$$

where a_j are constant coefficients and ω represents a specific sample path. We then apply regression to table 4 and obtain the following conditional expectation function $\mathbf{E}[Y|X] = -1.07 + 2.983X - 1.813X^2$. With the conditional expectation function as defined above, we are now able to calculate the value for continuation at time 2 and compare it to the exercise value of the option, also known as the intrinsic value of the option, at time two.

Table 7.4: Optimal strategy on exercise at time two

Path	Exercise value	Continuation
1	$(1.10-1.08)^+$=0.02	0.0369
2	$(1.10-1.26)^+$=0.00	-
3	$(1.10-1.07)^+$=0.03	0.0461
4	$(1.10-0.97)^+$=0.13	0.1176
5	$(1.10-1.56)^+$=0.00	-
6	$(1.10-0.77)^+$=0.33	0.152
7	$(1.10-0.84)^+$=0.26	0.1565
8	$(1.10-1.22)^+$=0.00	-

We find the continuation value by substituting a value for X into the conditional expectation formula. The comparison in table five yields that the optimal time to exercise would be to exercise the option on paths 4,6, and 7 at time two. The intrinsic value of the option is larger than the continuation value and indicates that early exercise would be optimal. These results lead to the following cash flow table at time 2, under the assumption that the option holder did not exercise the option prior to time 2.

Table 7.5: Cash flow table at time 2

Path	t=1	t=2	t=3
1	-	0.00	0.00
2	-	0.00	0.00
3	-	0.00	0.07
4	-	0.13	0.00
5	-	0.00	0.00
6	-	0.33	0.00
7	-	0.26	0.00
8	-	0.00	0.00

We observe that if the option is exercised at time two, it will result that the cash flow in the third column, or at time three, becomes zero. Once it is optimal to exercise the option and it is exercised, no further cash flows will occur because the option can only be exercised once. Hence proceeding with the algorithm, the next stage would be to determine whether the option will be exercised at time 1 or not.

The stock price table shows that there are five paths that will be in the money at time one. For the five paths that are in the money at time 1, we define as we did before for the discrete time two. The variable Y will denote the discounted value of the option's cash flows obtained from time two. Note that when we define Y, we will use actual realized cash flows from each possible path. Therefore we do not use the conditional expectation function for Y,

which we defined in time two, at time 1. So, the conditional expectation functions will differ for time 1 and time 2.

Since, an option can only be exercised once, we have that future cash flows can either occur at time two or time three, but are mutually exclusive. Therefore cash flows that realized in time two will be discounted back to time one and cash flows that realized in time three will also be discounted back to time one. As before, we define X, which represents the stock prices that were in the money at time one . So, we set up the following table for X and Y at time one,

Table 7.6: Regressions for time 1

Path	Y	X
1	0.00x0.94176	1.09
2	-	-
3	-	-
4	0.13x0.94176	0.93
5	-	-
6	0.33x0.94176	0.76
7	0.26x0.94176	0.92
8	0.00x0.94177	0.88

Once again, we need to calculate the conditional expectation function at time 1. This regression takes place by implementing the weighted Laguerre polynomials by regressing Y on a constant X and X^2. Thus, we are able to get the conditional expectation function for time 1, which is $\mathbf{E}[Y|X] = 2.038 - 3.335X + 1.356X^2$.

Now, substituting the value that we obtained for X, in table 7, into the conditional expectation function will yield continuation values at time 1. These continuation and exercise values are given in the table below and comparing these two values at time one is shown to be optimal for paths one, four, six, seven, and eight. Hence, the estimated values follows,

Table 7.7: Early exercise decision at time 1

Path	Exercise value	Continuation
1	$(1.10 - 1.09)^+$=0.01	0.0139
2	$(1.10 - 1.16)^+$=0.00	-
3	$(1.10 - 1.22)^+$=0.03	-
4	$(1.10 - 0.93)^+$=0.17	0.1092
5	$(1.10 - 1.11)^+$=0.00	-
6	$(1.10 - 0.76)^+$=0.34	0.2866
7	$(1.10 - 0.92)^+$=0.18	0.1175
8	$(1.10 - 0.88)^+$=0.22	0.1533

We have identified the optimal exercise strategy for all three discrete time steps. Therefore, we set up an optimal stopping table where one, in the table, indicates it would be optimal to early exercise the option.

Table 7.8: Optimally formulated stopping rule

Path	t=1	t=2	t=3
1	0	0	0
2	0	0	0
3	0	0	1
4	1	0	0
5	0	0	0
6	1	0	0
7	1	0	0
8	1	0	0

From this stopping rule table, it is logical to see where the option can be optimally exercised and it is intuitive to determine the cash flows that will realize for this option. The stopping rule table indicates at which time and in which path it will be optimal to exercise the option early. This leads to

the following option cash flow matrix,

Table 7.9: Option cash flows

Path	t=1	t=2	t=3
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.00	0.00	0.07
4	0.17	0.00	0.00
5	0.00	0.00	0.00
6	0.34	0.00	0.00
7	0.18	0.00	0.00
8	0.22	0.00	0.00

We have thus identified certain cash flows that were generated by the American put option. The cash flows were identified at each discrete time step along each path. The option value can now be calculated by discounting each value in the cash flow table, table 10. We discounted the cash flow values back to time zero. To obtain the simulated estimate for the option, we then sum over all the discounted values and divide it by the total number of paths, hence averaging over the paths. By applying this procedure we obtain the price for an American put option, which is 0.115. By comparison, it is almost twice as expensive as an European option (0.0564) with the same underlying stock prices.

The LSM or Least squares method algorithm illustrated how cross sectional information can be utilized in the simulation of paths to estimate the conditional expectation function. The conditional expectation function, however, is used to identify the exercise decision that maximizes the value of the option at each discrete time increment and along each simulated path.

Table 7.10: Discounted path values over the cash flow table

Path	t=0
1	0.000
2	0.000
3	0.066
4	0.160
5	0.000
6	0.320
7	0.170
8	0.207
Sum=	0.923
average=	0.115

The paper by Longstaff and Schwartz [39] shows the least squares Monte Carlo method is applied effectively, because only simple regression is applied.

The valuation framework

With the help of Longstaff and Schwartz [39], we create the following valuation framework for the LSM. We assume an underlying complete probability space $(\Omega, \mathcal{F}, \mathcal{P})$ that is defined over a time horizon of $[0, T]$, where T denotes the expiry of the contract. The state space Ω is defined as the set of all possible realizations, or paths, of the underlying process between $[0, T]$. Where ω represents a specific path, $\mathcal{F}$ represents a sigma field of distinguishable events at time T, and $\mathcal{P}$ is the probability measure that is defined on $\mathcal{F}$. Then $\mathbf{F}$ is defined as the filtration of the relevant underlying process, such that $\mathbf{F} = \{\mathcal{F}_t; t \in [0, T]\}$ and we assume that $\mathcal{F}_t = \mathcal{F}$. We also assume the existence of the risk neutral measure $\mathcal{Q}$, which leads to an arbitrage free pricing space.

We define the notation of the cash flows by $C(\omega, s; t, T)$, which represents a specific path ω generated by the option. This is dependent on the fact

that the optimal stopping rule, $s, t < s \leq T$, is followed and the option is not exercised early, prior to time t. The main objective of the least squares Monte Carlo algorithm is to obtain an optimal stopping rule that maximizes the value of an American option, in a path wise manner. To evaluate this American type option, we divide the interval $[0, T]$ into K discrete time steps of equal length. Hence, $0 < t_1 \leq t_2 \leq \cdots \leq t_K = T$ and we consider our optimal stopping rule at each of these points. If the option is specified in such a way that it is continuously exercisable, then we should set K a sufficiently large value. According to our optimal stopping rule, if the option is in the money at expiry then the option will be exercised. If, however, the option is out of the money at expiry, then we allow the option to expire worthless. At any other time, t_k, than expiry, the option holder has the choice either to exercise the option immediately or to continue the life of the option and revisit the exercise decision at the next exercise date. Hence, the value of the option is maximized; pathwise and logic would dictate that one should exercise the option when the immediate exercise is greater than, or equal to, the value of continuation.

During any time t_k, the cash flow of immediate exercise [39] of the option is known to the investor, the immediate exercise value is equal to the cash flow at this point in time. However, the cash flows generated by the continuation values are not known at any certain point in time t_k. According to the no arbitrage argument, the value of continuation, or the value of the option, assuming that it cannot be exercised until after t_k, is given by the remaining discounted cash flows $C(\omega, s; t_k, T)$ with respect to the risk neutral pricing measure $\mathcal{Q}$. We denote the value of continuation as $\mathbf{F}(\omega, t_k)$ at time t_k as,

$$\mathbf{F}(\omega; t_k) = \mathbf{E}_{\mathcal{Q}}[\sum_{j=k+1}^{K} e^{-\int_{t_k}^{t_j} r(\omega,s)ds} C(\omega, t_j; t_k, T)|\mathcal{F}_{t_k}] \qquad \text{(eq:7.1.1)}$$

where $r(\omega, s)$ denotes the risk-free interest rate which can possibly be stochastic in nature. The expectation is taken conditional on the filtration $\mathcal{F}_{t_k}$. Then

the problem breaks down into evaluating the immediate exercise value with this conditional expectation value and the optimal strategy is to exercise the option as soon as the immediate exercise value is greater than, or equal to, the conditional expectation value.

7.1 The Least Squares Monte Carlo algorithm

From the numerical example in the previous section [39], we saw that the least squares Monte Carlo method makes use of least square to approximate the conditional expectation function at discrete time steps $t_{K-1}, t_{K-2}, ..., t_1$. We find the paths of the cash flows with backwards recursion; for example, $C(\omega, s; t_k, T)$ may differ significantly from $C(\omega, s; t_{k+1}, T)$ since it may be optimal to stop and exercise the option at time t_{k+1}, therefore changing all the cash flows in a certain path, ω. If we specifically observe at time t_{K-1}, the unknown functional form of $\mathbf{F}(\omega; t_{K-1})$, as in equation 7.1.1 can be represented as a linear combination of a countable set of $\mathcal{F}_{t_{K-1}}$-measurable basis functions.

The $\mathcal{F}_{t_{K-1}}$ has measurable basis functions, if the conditional expectation function is an element of the L^2 space or Hilbert space of squared integrable functions relative to some measure. The L^2 space has a finite number of orthonormal basis. Then we are able to represent the conditional expectation as a linear function of the basis elements.

We define X to be the value of the underlying asset and we assume that X follows a Markov process. Some choices of basis functions are as follows,

Laguerre polynomials

$$
\begin{aligned}
L_0(X) &= 1 \\
L_1(X) &= (1 - X) \\
L_2(X) &= (1 - 2X + \frac{X^2}{2}) \\
&\cdots \\
L_n(X) &= \frac{e^X}{n!}\frac{d^n}{dX^n}(X^n e^{-X})
\end{aligned}
$$

Weighted Laguerre polynomials

$$
\begin{aligned}
L_0(X) &= e^{\frac{-X}{2}} \\
L_1(X) &= e^{\frac{-X}{2}}(1 - X) \\
L_2(X) &= e^{\frac{-X}{2}}(1 - 2X + \frac{X^2}{2}) \\
&\cdots \\
L_n(X) &= e^{\frac{-X}{2}}\frac{e^X}{n!}\frac{d^n}{dX^n}(X^n e^{-X})
\end{aligned}
$$

Monomial basis functions

$$
\begin{aligned}
L_0(X) &= 1 \\
L_1(X) &= X \\
L_2(X) &= X^2 \\
&\cdots \\
L_n(X) &= X^n
\end{aligned}
$$

Hermite polynomials

$$
\begin{aligned}
&L_0(X) = 1 \\
&L_1(X) = 2X \\
&L_2(X) = 4X^2 - 2 \\
&\quad \dots \\
&L_n(X) = (-1)^n e^{t^2} \frac{d^n e^{-t^2}}{dt^n}, \qquad n = 0, 1, 2, ... \, and \, -\infty < t < \infty
\end{aligned}
$$

Then the value of continuation, $\mathbf{F}(\omega; t_{K-1})$ can be defined as,

$$\mathbf{F}(\omega; t_{K-1}) = \sum_{j=0}^{\infty} a_j L_j(X),$$

where a_j are constant coefficients. Other types of basis functions include Jacobi polynomials, Gegenbauer polynomials, Legendre polynomials, and Chebyshev polynomials.

Therefore, to find an approximation for $F(\omega; t_{K-1})$ by implementing the least squares Monte Carlo approach, we use the first $M < \infty$ basis functions. We denote this approximation as $F_M(\omega; t_{K-1})$. We therefore estimate $F_M(\omega; t_{K-1})$ by regressing the discounted values $C(\omega, s; t_{k-1}, T)$ onto the specified basis functions. But this is only done for in the money paths at time t_{K-1}. We use these in the money paths for the estimation, since the optimal decision dictates that the option will be exercised only when the option is in the money. Since we are only focusing on the in the money paths for the conditional expectation, far less basis functions are required to obtain an accurate approximation.

Chapter 8

Results for American Puts

The results below are similar to the results obtained by Longstaff and Schwartz [2001] paper, with Strike price of 40 and where the option is exercisable 50 times per year. The European option values are based on the closed form Black-Scholes formula. The risk-free interest rate is assumed to be 6%, S denotes the underlying stock price, and σ denotes the volatility of returns. The number of years to expiration is two. The simulation is based on 100 000 paths of the stock price process, while implementing antithetic variates. (s.e.) denotes the standard estimation error.

Table 8.1: Longstaff and Schwartz results

S_0	T	σ	Closed form EU	Bin Tree	American put	(s.e.)
36	1	0.2	3.844	4.476	4.471	(0.010)
36	2	0.2	3.763	4.841	4.820	(0.012)
36	1	0.4	6.711	7.102	7.092	(0.020)
36	2	0.4	7.7	8.510	8.489	(0.024)
38	1	0.2	2.852	3.252	3.242	(0.009)
38	2	0.2	2.991	3.748	3.737	(0.011)
38	1	0.4	5.834	6.145	6.137	(0.019)
38	2	0.4	6.979	7.672	7.668	(0.022)
40	1	0.2	2.066	2.313	2.313	(0.009)
40	2	0.2	2.356	2.883	2.876	(0.010)
40	1	0.4	5.06	5.311	5.306	(0.018)
40	2	0.4	6.326	6.923	6.921	(0.022)
42	1	0.2	1.465	1.618	1.616	(0.007)
42	2	0.2	1.841	2.212	2.209	(0.010)
42	1	0.4	4.379	4.581	4.581	(0.017)
42	2	0.4	5.736	6.247	6.243	(0.021)
44	1	0.2	1.017	1.111	1.116	(0.007)
44	2	0.2	1.429	1.691	1.675	(0.009)
44	1	0.4	3.783	3.946	3.955	(0.017)
44	2	0.4	5.202	5.648	5.622	(0.021)

8.1 Results for Weighted Laguerre polynomials approximation

Table 8.2: Error analysis for Weighted Laguerre American put options

	Halton	Sobol	Faure	CG
Max Error	0.3534	0.0438	0.1676	0.105
Min Error	0.0005	0.0036	0.0007	0.0403
Error stdev	0.092314	0.013764	0.06051	0.065008
% avg est error	2.881%	0.573%	1.463%	1.928%
Avg CPU time	0.250785	0.258615	0.22502	0.04378

Longstaff and Schwartz used the same weighted Laguerre basis functions for their article. They found the minimum and maximum approximation error to be 0.7-2.4 cents, respectively, on 100 000 simulations using Antithetic variates and 50 different exercise points per year. In our approximations we used only 1 000 simulations, implementing antithetic variates, for three different Low discrepancy sequences and a Pseudo random generated sequence, with 50 different exercise points per year. Our aim was to cut on the computational cost of the calculations and still achieve good accuracy.

For the Halton sequence, we obtained a minimum and maximum error of 0.05-35.34 cents, respectively. With a standard error deviation of 9 cents and an average approximation error of 2.881%. The average computational time used to calculate the approximation was 0.250785 seconds.

From our results in table 10.2, we observe that the Sobol sequence performed exceptionally well with the smallest average approximation error of 0.573%. Compared to the other sequences, the Sobol sequence yielded the best results. Low discrepancy sequences are, however, computationally costly, with the average time taken to compute these approximations being 5-6 times longer compared to the Pseudo random sequences.

Table 8.3: European and American put prices by LSM

			Halton		Sobol		Faure		CG	
S_0	T	σ	Euro	American	Euro	American	Euro	American	Euro	American
36	1	0.2	3.7563	4.7171	3.9274	4.4452	3.7107	4.4673	4.0875	4.2409
36	2	0.2	3.7127	4.9306	3.8642	4.8326	3.8763	4.9357	4.2052	4.7426
36	1	0.4	6.4189	7.1777	6.4944	7.0501	6.9269	7.056	6.9536	6.9629
36	2	0.4	7.6007	8.5888	7.4527	8.4938	8.095	8.3481	8.8713	8.383
38	1	0.2	2.9134	3.3333	2.6405	3.2232	2.7267	3.2497	2.893	3.2037
38	2	0.2	2.9882	3.8322	3.0355	3.7787	2.9558	3.7388	3.3279	3.6119
38	1	0.4	5.7591	6.2015	5.8718	6.1759	5.66	6.0353	6.3192	6.1599
38	2	0.4	6.6519	7.3156	6.8358	7.6252	6.8995	7.5251	7.7961	7.4698
40	1	0.2	2.1055	2.4394	2.054	2.3492	2.1578	2.3255	2.2365	2.3139
40	2	0.2	2.3063	2.9534	2.3547	2.8913	2.1847	2.7844	2.5094	2.8236
40	1	0.4	4.6869	5.1674	4.7074	5.3159	4.8535	5.4756	5.2745	5.193
40	2	0.4	6.5442	7.2047	6.3097	6.9531	6.4502	6.8895	7.0901	6.9641
42	1	0.2	1.5957	1.7253	1.3914	1.6134	1.4701	1.6099	1.5518	1.5877
42	2	0.2	1.8265	2.3702	1.8694	2.2229	1.8036	2.1792	2.1443	2.2498
42	1	0.4	4.2697	4.6037	4.2003	4.5642	4.3317	4.5887	4.5652	4.6297
42	2	0.4	5.758	6.22	5.6101	6.2284	5.5742	6.09	6.2582	6.0667
44	1	0.2	1.0288	1.1831	1.1186	1.1371	1.0877	1.1452	1.0799	1.0556
44	2	0.2	1.4472	1.6755	1.2835	1.6708	1.4229	1.6012	1.5783	1.6845
44	1	0.4	3.7686	3.9406	3.8494	3.9814	3.7569	3.9586	4.1475	3.9851
44	2	0.4	5.1833	5.6537	5.1843	5.6183	5.3282	5.49	6.0614	5.5037

[The above results are obtained for both European and American put options implementing the Weighted Laguerre basis functions, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 3 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year.].

Table 8.4: Weighted Laguerre respective errors

			Halton		Sobol		Faure		CG	
S_0	T	σ	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error
36	1	0.2	0.0841	0.2451	0.0755	0.0268	0.0800	0.0047	0.0853	0.2311
36	2	0.2	0.0884	0.1096	0.0848	0.0116	0.0944	0.1147	0.0932	0.0784
36	1	0.4	0.1406	0.0867	0.1353	0.0409	0.1363	0.0350	0.1322	0.1281
36	2	0.4	0.1754	0.1008	0.1576	0.0058	0.1727	0.1399	0.1633	0.1050
38	1	0.2	0.0845	0.0893	0.0807	0.0208	0.0836	0.0057	0.0937	0.0403
38	2	0.2	0.1002	0.0972	0.1042	0.0437	0.1000	0.0038	0.0953	0.1231
38	1	0.4	0.1396	0.0625	0.1381	0.0369	0.1321	0.1037	0.1387	0.0209
38	2	0.4	0.1642	0.3534	0.1650	0.0438	0.1700	0.1439	0.1661	0.1992
40	1	0.2	0.0860	0.1264	0.0797	0.0362	0.0807	0.0125	0.0735	0.0009
40	2	0.2	0.0986	0.0744	0.0971	0.0123	0.0956	0.0946	0.0939	0.0554
40	1	0.4	0.1372	0.1406	0.1397	0.0079	0.1365	0.1676	0.1365	0.1150
40	2	0.4	0.1633	0.2837	0.1669	0.0321	0.1651	0.0315	0.1631	0.0431
42	1	0.2	0.0745	0.1083	0.0694	0.0036	0.0747	0.0071	0.0724	0.0293
42	2	0.2	0.0893	0.1642	0.0917	0.0169	0.0858	0.0268	0.0921	0.0438
42	1	0.4	0.1340	0.0157	0.1450	0.0238	0.1368	0.0007	0.1315	0.0417
42	2	0.4	0.1608	0.0230	0.1575	0.0146	0.1664	0.1530	0.1636	0.1763
44	1	0.2	0.0607	0.0651	0.0561	0.0191	0.0633	0.0272	0.0649	0.0624
44	2	0.2	0.0820	0.0005	0.0812	0.0042	0.0817	0.0738	0.0845	0.0095
44	1	0.4	0.1363	0.0164	0.1257	0.0244	0.1270	0.0016	0.1411	0.0281
44	2	0.4	0.1513	0.0317	0.1587	0.0037	0.1635	0.1320	0.1695	0.1183

[The above results are obtained for American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility, which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 2 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year. Error indicates an approximation error for the simulated price relative to the Longstaff and Schwartz [2001] results. $(s.e.)$ denotes the standard theoretical estimation error.].

8.2 Results for Laguerre polynomials approximation

Table 8.5: Error Analysis for Laguerre approximation

	Halton	Sobol	Faure	CG
Max Error	0.1907	0.1039	0.3979	0.0043
Min Error	0.0118	0.0004	0.0009	0.0292
Error stdev	0.052561	0.026392	0.093159	0.09988
% avg est error	1.267%	0.806%	1.382%	2.896%
Avg CPU time	0.260965	0.267205	0.254695	0.089875

As mentioned before, the Longstaff and Schwartz used weighted Laguerre basis functions for their article. They found the minimum and maximum approximation error to be 0.7-2.4 cents, respectively, on 100 000 simulations using Antithetic variates and 50 different exercise points per year.
In our Laguerre approximations we used only 1 000 simulations, implementing antithetic variates, for three different Low discrepancy sequences and a Pseudo random generated sequence, with 50 different exercise points per year.
For the Sobol sequence we obtained a minimum and maximum error of 0.04-10.39 cents, respectively. With a standard error deviation of 2.6392 cents and an average approximation error of 0.806%. The average computational time used to calculate the approximation was 0.267205 seconds.
From our results in table 8.5, we observe that the Sobol sequence performed exceptionally well with the smallest average approximation error of 0.806%. Compared to the other sequences, the Sobol sequence yielded the most accurate results. The Low discrepancy sequences were, however, computationally costly, with the average time taken to compute these approximations being 3-4 times longer compared to the Pseudo random sequences.

Table 8.6: European and American put prices by LSM simulation on Laguerre Polynomial basis functions

			Halton		Sobol		Faure		CG	
S_0	T	σ	Euro	American	Euro	American	Euro	American	Euro	American
36	1	0.2	3.7761	4.6222	3.6951	4.4377	3.7156	4.5473	3.8385	4.5022
36	2	0.2	3.8188	4.945	3.6776	4.8142	3.8214	4.8799	3.7674	4.8581
36	1	0.4	6.528	7.04	6.5483	7.1092	6.5987	7.0845	6.3358	6.9172
36	2	0.4	7.3627	8.4762	7.8368	8.5919	7.9114	8.8859	7.6693	8.5123
38	1	0.2	2.7343	3.3487	2.8614	3.2685	2.8546	3.2567	2.8599	3.2792
38	2	0.2	3.0903	3.7122	2.8769	3.7686	2.9823	3.713	2.9271	3.9364
38	1	0.4	5.7449	6.1908	5.8414	6.193	5.8559	6.0974	5.9868	6.4985
38	2	0.4	7.0377	7.754	6.803	7.6432	7.0337	7.7781	7.0215	7.5975
40	1	0.2	1.9532	2.2768	2.0415	2.3049	1.946	2.3382	2.1046	2.3661
40	2	0.2	2.1764	2.7674	2.4276	2.8745	2.3056	2.9021	2.3987	3.0295
40	1	0.4	4.7298	5.3315	4.7782	5.3558	5.1287	5.3809	5.0103	5.3306
40	2	0.4	6.1201	7.0526	6.21	6.9219	6.338	6.8224	6.4317	7.1993
42	1	0.2	1.5221	1.6977	1.4978	1.6498	1.4209	1.6179	1.5297	1.6884
42	2	0.2	1.9612	2.3967	1.8548	2.2859	1.8928	2.2373	2.0424	2.3781
42	1	0.4	4.2394	4.4057	4.3782	4.5697	4.5671	4.8104	4.2512	4.5566
42	2	0.4	5.4164	6.1125	5.5997	6.2151	5.6175	6.3164	5.5611	6.253
44	1	0.2	0.9726	1.1656	0.9205	1.1176	0.9652	1.1069	1.0696	1.1661
44	2	0.2	1.5812	1.7568	1.2908	1.7141	1.3768	1.6658	1.3972	1.8807
44	1	0.4	3.8297	4.0817	3.4724	3.9688	3.6947	3.9184	3.7393	3.9573
44	2	0.4	5.0104	5.5618	4.9308	5.607	5.3349	5.7441	5.1481	5.602

[The above results are obtained for both European and American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 3 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year.].

Table 8.7: Error analysis for respective Laguerre Approximations

			Halton		Sobol		Faure		CG	
S_0	T	σ	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error
36	1	0.2	0.0862	0.1502	0.0945	0.0343	0.0907	0.0753	0.1067	0.0242
36	2	0.2	0.1264	0.1240	0.1088	0.0068	0.1058	0.0589	0.1153	0.0181
36	1	0.4	0.1906	0.0510	0.1812	0.0182	0.2046	0.0065	0.2069	0.1838
36	2	0.4	0.2384	0.0118	0.2283	0.1039	0.2260	0.3979	0.2286	0.0043
38	1	0.2	0.0974	0.1047	0.0924	0.0245	0.1051	0.0127	0.0928	0.0292
38	2	0.2	0.1125	0.0228	0.1056	0.0336	0.1136	0.0220	0.1035	0.1914
38	1	0.4	0.2150	0.0518	0.1943	0.0540	0.1997	0.0416	0.1881	0.3505
38	2	0.4	0.2260	0.0850	0.2367	0.0258	0.2187	0.1091	0.2123	0.0725
40	1	0.2	0.0800	0.0362	0.0789	0.0081	0.0891	0.0252	0.0856	0.0521
40	2	0.2	0.1076	0.1116	0.1218	0.0045	0.1026	0.0231	0.1115	0.1445
40	1	0.4	0.1824	0.0235	0.1766	0.0478	0.1740	0.0729	0.1708	0.0186
40	2	0.4	0.2284	0.1316	0.2157	0.0009	0.2232	0.0986	0.2180	0.2793
42	1	0.2	0.0754	0.0807	0.0721	0.0328	0.0772	0.0009	0.0716	0.0714
42	2	0.2	0.0884	0.1907	0.1137	0.0799	0.1071	0.0313	0.0999	0.1661
42	1	0.4	0.1757	0.1823	0.1724	0.0183	0.1754	0.2224	0.1726	0.0254
42	2	0.4	0.2184	0.1305	0.2104	0.0279	0.2224	0.0734	0.2186	0.0050
44	1	0.2	0.0704	0.0476	0.0654	0.0004	0.0722	0.0111	0.0681	0.0561
44	2	0.2	0.0862	0.0818	0.0917	0.0391	0.0932	0.0092	0.0848	0.1907
44	1	0.4	0.1583	0.1247	0.1601	0.0118	0.1709	0.0386	0.1612	0.0093
44	2	0.4	0.1907	0.0602	0.2069	0.0150	0.2029	0.1221	0.2095	0.0450

[The above results are obtained for American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility, which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 2 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year. Error indicates an approximation error for the simulated price relative to the Longstaff and Schwartz [2001] results. $(s.e.)$ denotes the standard theoretical estimation error.].

8.3 Results for Monomial polynomials approximation

Table 8.8: Error Analysis for Monomial approximation
Error Aanlysis

	Halton	Sobol	Faure	CG
Max Error	0.1684	0.1229	0.370783	0.226
Min Error	0.0182	0.0043	0.006312	0.1093
Error stdev	0.039232	0.026314	0.091495	0.061766
% avg est error	0.4468%	1.1137%	3.0857%	2.5687%
Avg CPU time	0.214855	0.25002	0.133594	0.084405

As mentioned before, the Longstaff and Schwartz used the Weighted Laguerre basis functions for their article. They found the minimum and maximum approximation error to be 0.7-2.4 cents, respectively, on 100 000 simulations using Antithetic variates and 50 different exercise points per year. In our Monomial approximations we used only 1 000 simulations, implementing antithetic variates, for three different Low discrepancy sequences and a Pseudo random generated sequence, with 50 different exercise points per year. For the Faure sequence we obtained a minimum and maximum error of 0.6312 - 17.0783 cents, respectively. With a standard error deviation of 9.1495 cents and an average approximation error of 3.0857%. The average computational time used to calculate the approximation was 0.133594 seconds. From our results in table 8.8, we observe that the Halton sequence performed exceptionally well, with the smallest average approximation error of 0.4468%. Compared to the other sequences, the Halton sequence yielded the most accurate results. The Low discrepancy sequences were, however computationally costly, with the average time taken to compute these approximations being 1.5-3 times longer compared to the Pseudo random sequences.

Table 8.9: European and American put prices by LSM simulation on Monomial Polynomial basis functions

			Halton		Sobol		Faure		CG	
S0	T	sigma	Euro	American	Euro	American	Euro	American	Euro	American
36	1	0.2	3.7522	4.5152	3.9172	4.4954	3.8999	4.4955	3.8734	4.5303
36	2	0.2	3.6700	4.7636	3.7802	4.8757	3.5311	4.7011	3.8269	4.9291
36	1	0.4	6.5663	7.2307	6.7511	7.1280	6.8317	7.3071	6.7160	7.1412
36	2	0.4	7.6688	8.5302	7.6939	8.5259	7.6199	8.4431	7.5774	8.7340
38	1	0.2	2.7130	3.3315	2.7146	3.2360	2.9646	3.3150	2.9232	3.3593
38	2	0.2	2.8405	3.7834	3.0560	3.7307	3.0477	3.8175	2.9056	3.7715
38	1	0.4	5.8502	6.1951	5.5597	6.1088	5.9814	6.2303	5.9086	6.0889
38	2	0.4	7.1415	7.7906	6.6499	7.6202	6.5907	7.5786	6.8614	7.7512
40	1	0.2	2.0006	2.3666	2.1769	2.3812	2.1153	2.3656	1.9587	2.2785
40	2	0.2	2.1506	2.8443	2.5601	2.8632	2.5110	2.9409	2.3439	2.9406
40	1	0.4	4.9547	5.2898	4.8408	5.2681	5.1973	5.4810	4.9547	5.2753
40	2	0.4	6.2087	6.8598	6.2521	6.9482	6.3549	7.1022	6.1249	7.1234
42	1	0.2	1.4455	1.6921	1.4101	1.6432	1.4617	1.7201	1.3477	1.6639
42	2	0.2	1.6979	2.1749	1.8802	2.2674	1.8885	2.2461	2.0109	2.3379
42	1	0.4	4.4325	4.6670	4.1926	4.4651	4.2869	4.7280	4.4094	4.5146
42	2	0.4	5.6410	6.3354	5.7620	6.1913	6.0849	6.6138	5.6587	6.4196
44	1	0.2	0.8977	1.0703	0.9802	1.1459	1.1397	1.2370	1.1079	1.2775
44	2	0.2	1.4635	1.6935	1.2281	1.7075	1.3521	1.6687	1.3696	1.6955
44	1	0.4	3.6210	3.7886	3.8195	3.9698	3.7356	4.2558	3.6677	4.0148
44	2	0.4	5.1527	5.6949	5.1230	5.5744	4.9383	5.5091	5.2910	5.7654

[The above results are obtained for both European and American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 3 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year.].

Table 8.10: Error analysis for respective Monomial Approximations

			Halton		Sobol		Faure		CG	
S_0	T	σ	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error
36	1	0.2	0.0957	0.0432	0.0937	0.0234	0.1052	0.0234	0.0909	0.0523
36	2	0.2	0.1029	0.0574	0.1207	0.0547	0.1167	0.0547	0.1169	0.0891
36	1	0.4	0.1920	0.1397	0.1888	0.0370	0.1791	0.0370	0.1879	0.0402
36	2	0.4	0.2080	0.0422	0.2380	0.0379	0.2327	0.0379	0.2267	0.2260
38	1	0.2	0.0982	0.0875	0.1012	0.0080	0.0936	0.0080	0.0880	0.1093
38	2	0.2	0.1129	0.0484	0.1055	0.0043	0.1197	0.0043	0.1048	0.0265
38	1	0.4	0.1890	0.0561	0.1945	0.0302	0.1841	0.0302	0.1860	0.0591
38	2	0.4	0.2102	0.1216	0.2151	0.0488	0.2127	0.0488	0.2291	0.0812
40	1	0.2	0.0853	0.0536	0.0896	0.0682	0.0904	0.0682	0.0878	0.0355
40	2	0.2	0.0999	0.0347	0.1026	0.0158	0.1062	0.0158	0.1139	0.0556
40	1	0.4	0.1854	0.0182	0.1781	0.0399	0.1985	0.0399	0.1706	0.0367
40	2	0.4	0.2275	0.0612	0.2222	0.0272	0.2312	0.0272	0.2170	0.2034
42	1	0.2	0.0779	0.0751	0.0759	0.0262	0.0803	0.0262	0.0793	0.0469
42	2	0.2	0.0990	0.0311	0.0977	0.0614	0.1019	0.0614	0.0974	0.1259
42	1	0.4	0.1846	0.0790	0.1738	0.1229	0.1711	0.1229	0.1664	0.0674
42	2	0.4	0.2143	0.0924	0.2145	0.0517	0.2295	0.0517	0.2051	0.1716
44	1	0.2	0.0616	0.0477	0.0612	0.0279	0.0705	0.0279	0.0744	0.1675
44	2	0.2	0.0957	0.0185	0.0975	0.0325	0.0943	0.0325	0.0908	0.0055
44	1	0.4	0.1719	0.1684	0.1667	0.0128	0.1709	0.0128	0.1640	0.0668
44	2	0.4	0.2126	0.0729	0.2003	0.0476	0.1993	0.0476	0.2077	0.1184

[The above results are obtained for American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 2 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year. Error indicates an approximation error for the simulated price relative to the Longstaff and Schwartz [2001] results. $(s.e.)$ denotes the standard theoretical estimation error.].

8.4 Results for Hermite polynomials approximation

Table 8.11: Error Analysis for Hermite approximation

	Halton	Sobol	Faure	CG
Max Error	0.4969	0.2987	0.3596	0.3333
Min Error	0.0011	0.0239	0.0058	0.0034
Error stdev	0.0392	0.0263	0.0915	0.0618
% avg est error	3.849%	2.791%	2.719%	2.794%
Avg CPU time	1.0039	0.3063	0.0602	0.1273

As mentioned previously, the Longstaff and Schwartz used the Weighted Laguerre basis functions for their article. They found the minimum and maximum approximation error to be 0.7-2.4 cents, respectively, on 100 000 simulations using Antithetic variates and 50 different exercise points per year. In our Laguerre approximations we used only 1 000 simulations, implementing antithetic variates, for three different Low discrepancy sequences and a Pseudo random generated sequence, with 50 different exercise points per year. For the CG or computer generated (Pseudo random) sequence we obtained a minimum and maximum error of 0.34 - 33.33 cents, respectively. With a standard error deviation of 6.18 cents and an average approximation error of 2.794%. The average computational time used to calculate the approximation was 0.1273 seconds. From our results in table 22, we observe that all three low discrepancy and Pseudo random sequence obtained relatively large average estimation errors. These average errors are calculated over the 20 simulations as seen in table 8.11. The Low discrepancy sequences were, however, computationally costly, with the average time taken to compute these approximations being 2-10 times longer compared to the Pseudo random sequences.

Table 8.12: Results for Hermite polynomials approximation

			Halton		Sobol		Faure		CG	
S_0	T	σ	Euro	American	Euro	American	Euro	American	Euro	American
36	1	0.2	3.9413	4.4557	4.0586	4.5934	3.7824	4.4073	3.7689	4.3731
36	2	0.2	3.8845	4.9569	3.5603	4.8498	3.5302	4.8152	3.8613	4.8946
36	1	0.4	7.0299	7.4169	6.6056	7.0133	6.5138	6.9698	6.9467	7.4243
36	2	0.4	7.9011	8.8621	7.9119	8.5699	7.8961	8.7878	7.5374	8.3707
38	1	0.2	2.7850	3.3079	2.7772	3.2142	2.8938	3.4256	2.7602	3.2373
38	2	0.2	2.9668	3.7361	3.1910	3.8275	2.9672	3.7002	3.2120	3.8509
38	1	0.4	5.5235	5.8685	5.9069	6.4377	6.2420	6.4408	5.7079	6.1356
38	2	0.4	6.9140	7.4649	6.8983	7.7865	6.7704	7.7108	6.9586	7.6267
40	1	0.2	2.2729	2.4570	2.2474	2.4654	2.1088	2.3413	2.1827	2.4504
40	2	0.2	2.2108	2.8930	2.4337	2.9430	2.3825	2.8466	2.4389	2.9838
40	1	0.4	5.2683	5.8049	5.0788	5.4000	4.9198	5.1798	5.0096	5.4733
40	2	0.4	5.9905	6.6571	6.1441	7.0399	6.0857	6.8838	6.4108	7.0434
42	1	0.2	1.5392	1.7029	1.3664	1.5910	1.4299	1.5879	1.5190	1.6439
42	2	0.2	1.9089	2.2585	1.7577	2.2299	1.8273	2.3041	1.6896	2.1145
42	1	0.4	4.8189	4.9177	4.4490	4.6385	3.9470	4.3746	3.8341	4.2699
42	2	0.4	5.4329	6.1417	5.9739	6.4554	5.4904	6.1711	5.7591	6.4715
44	1	0.2	0.9744	1.0603	1.0424	1.2151	0.8960	1.0924	0.9734	1.0944
44	2	0.2	1.3952	1.7052	1.5346	1.8053	1.4812	1.7733	1.5158	1.7201
44	1	0.4	3.9871	4.2197	3.6898	3.9112	3.9114	4.1026	3.9731	4.1382
44	2	0.4	5.3941	5.9443	4.8791	5.4208	5.3143	5.9816	4.9074	5.5147

[The above results are obtained for both European and American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 3 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year.].

Table 8.13: Error analysis for respective Hermite Approximations

			Halton		Sobol		Faure		CG	
S_0	T	σ	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error
36	1	0.2	0.101	0.016	0.091	0.121	0.099	0.065	0.101	0.099
36	2	0.2	0.108	0.136	0.122	0.029	0.113	0.006	0.113	0.074
36	1	0.4	0.184	0.326	0.198	0.078	0.185	0.121	0.178	0.333
36	2	0.4	0.215	0.374	0.223	0.082	0.236	0.300	0.226	0.117
38	1	0.2	0.097	0.064	0.085	0.030	0.098	0.182	0.096	0.007
38	2	0.2	0.119	0.001	0.121	0.093	0.109	0.035	0.105	0.116
38	1	0.4	0.192	0.270	0.199	0.299	0.189	0.302	0.186	0.003
38	2	0.4	0.220	0.204	0.223	0.117	0.205	0.042	0.222	0.042
40	1	0.2	0.090	0.144	0.090	0.152	0.088	0.028	0.091	0.137
40	2	0.2	0.099	0.014	0.105	0.064	0.105	0.032	0.098	0.105
40	1	0.4	0.184	0.497	0.173	0.092	0.174	0.128	0.193	0.165
40	2	0.4	0.216	0.264	0.213	0.119	0.234	0.037	0.218	0.122
42	1	0.2	0.086	0.086	0.085	0.026	0.085	0.029	0.075	0.027
42	2	0.2	0.106	0.052	0.089	0.024	0.092	0.098	0.096	0.092
42	1	0.4	0.189	0.330	0.201	0.051	0.193	0.213	0.182	0.318
42	2	0.4	0.213	0.101	0.220	0.212	0.207	0.072	0.233	0.228
44	1	0.2	0.068	0.058	0.067	0.097	0.076	0.026	0.060	0.024
44	2	0.2	0.083	0.030	0.094	0.130	0.095	0.098	0.082	0.045
44	1	0.4	0.176	0.263	0.159	0.046	0.161	0.146	0.164	0.181
44	2	0.4	0.213	0.322	0.204	0.201	0.209	0.360	0.210	0.107

[The above results are obtained for American put options, using three different Low Discrepancy sequences. CG denotes computer generated and is our Psuedo random sequence. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 2 years. We base our results on 1 000 antithetic simulations, using 50 exercise points per year. Error indicates an approximation error for the simulated price relative to the Longstaff and Schwartz [2001] results. $(s.e.)$ denotes the standard theoretical estimation error.].

8.5 Results with Control variates and Weighted Laguerre polynomials approximation

Table 8.14: Error Analysis for Control variate approximation

	Halton	Sobol	Faure
Max Error	0.4286	0.4808	0.4793
Min Error	0.0041	0.0052	0.0002
Error stdev	0.1525	0.1683	0.1534
% avg est error	2.5984%	2.9105%	2.5938%
Avg CPU time	0.5508	0.5922	1.0766

We implemented the Pseudo random sequence as the control variate in the following approximation. In the control variate results we obtained +2.5% errors (from table 8.14) for all three low discrepancy sequences, compared to the Longstaff and Schwartz results. From table 8.16 we observe that in a relatively high volatility environment, the control variate approximations underestimate the price of the option. But in lower volatility environment the approximations are accurate.

In our control variate approximations, we used only 5 000 simulations, implementing antithetic variates, for three different low discrepancy sequences with 50 different exercise points per year.

The control variate approximations are also computationally more demanding using a minimum of 0.5508 seconds for the Halton approximate and a maximum of 1.0766 seconds for computing the Faure estimate. This is far more time that is necessary to obtain a approximate compared to the Antithetic method.

The method is applicable and deliver relatively good accuracy in low volatility situations, but should be avoided when working with relatively large volatilities.

Table 8.15: Results with control variates and weighted Laguerre polynomials approximation

			Halton		Sobol		Faure	
S_0	T	σ	Euro	American	Euro	American	Euro	American
36	1	0.2	3.8265	4.452	3.8972	4.4849	3.7528	4.4764
36	2	0.2	3.7698	4.8303	3.691	4.815	3.6414	4.8351
36	1	0.4	6.5797	6.7407	6.5949	6.8585	6.5581	6.8061
36	2	0.4	7.7889	8.127	7.8946	8.0072	7.8166	8.0087
38	1	0.2	2.8821	3.2481	2.9143	3.2241	2.8646	3.2442
38	2	0.2	3.0031	3.7403	3.0293	3.7206	2.9891	3.7452
38	1	0.4	5.781	5.8379	5.8333	5.8052	5.7853	5.8291
38	2	0.4	7.0211	7.2404	6.9924	7.2014	7.2753	7.29
40	1	0.2	2.1167	2.3221	2.0461	2.2943	2.089	2.3062
40	2	0.2	2.2888	2.8572	2.3881	2.8192	2.4172	2.8416
40	1	0.4	5.064	5.0914	5.015	5.0882	5.1531	5.07
40	2	0.4	6.4124	6.592	6.198	6.5572	6.2359	6.591
42	1	0.2	1.3966	1.6332	1.4819	1.6256	1.4822	1.6595
42	2	0.2	1.8616	2.2192	1.7891	2.2008	1.8342	2.2126
42	1	0.4	4.2135	4.4233	4.2852	4.3102	4.3511	4.3828
42	2	0.4	5.5795	6.0497	5.8121	5.9059	5.751	6.0059
44	1	0.2	0.9785	1.1229	1.0543	1.1026	0.9858	1.1074
44	2	0.2	1.3534	1.6538	1.4684	1.6969	1.4388	1.6787
44	1	0.4	3.7401	3.7102	3.7163	3.7882	3.7781	3.7536
44	2	0.4	5.1496	5.3319	5.2961	5.3485	5.358	5.4421

[The above results are obtained for both European and American put options, using three different Low Discrepancy sequences. Psuedo random sequence will be our control estimate. S_0 denotes the initial price of the underlying and σ denotes the volatility which fluctuates between 0.2 and 0.4. Time to maturity is denoted by T and changes between 1 and 2 years. We base our results on 5 000 antithetic simulations, using 50 exercise points per year.].

Table 8.16: Errors for respective control variate Approximations

			Halton		Sobol		Faure	
S_0	T	σ	(s.e.)	Error	(s.e.)	Error	(s.e.)	Error
36	1	0.2	0.0186	0.02	0.0192	0.0129	0.0186	0.0044
36	2	0.2	0.021	0.0093	0.0228	0.006	0.0207	0.0141
36	1	0.4	0.0353	0.3503	0.035	0.2325	0.0331	0.2849
36	2	0.4	0.0387	0.361	0.0374	0.4808	0.0355	0.4793
38	1	0.2	0.0215	0.0041	0.0221	0.0199	0.0235	0.0002
38	2	0.2	0.0244	0.0053	0.0233	0.0144	0.0237	0.0102
38	1	0.4	0.0337	0.3011	0.0326	0.3338	0.0358	0.3099
38	2	0.4	0.0378	0.4286	0.0392	0.4676	0.0403	0.379
40	1	0.2	0.0232	0.0091	0.0269	0.0187	0.0223	0.0068
40	2	0.2	0.0202	0.0218	0.0217	0.0598	0.0242	0.0374
40	1	0.4	0.0384	0.2166	0.0343	0.2198	0.0335	0.238
40	2	0.4	0.0412	0.329	0.041	0.3638	0.0405	0.33
42	1	0.2	0.0224	0.0162	0.0229	0.0086	0.0232	0.0425
42	2	0.2	0.024	0.0132	0.0238	0.0052	0.0234	0.0066
42	1	0.4	0.033	0.1647	0.038	0.2778	0.0318	0.2052
42	2	0.4	0.0343	0.1933	0.0357	0.3371	0.0375	0.2371
44	1	0.2	0.0254	0.0049	0.0237	0.0154	0.0234	0.0106
44	2	0.2	0.0231	0.0212	0.0219	0.0219	0.0241	0.0037
44	1	0.4	0.0332	0.2468	0.0353	0.1688	0.034	0.2034
44	2	0.4	0.0424	0.2901	0.0398	0.2735	0.038	0.1799

[The above results are obtained for American put options, using three different Low Discrepancy sequences. S_0 denotes the initial price of the underlying and σ denotes the volatility. Time to maturity is denoted by T. We base our results on 5 000 antithetic simulations, using 50 exercise points per year. Error indicates an approximation error for the simulated price relative to the Longstaff and Schwartz [2001] results. $(s.e.)$ denotes the standard theoretical estimation error.].

Chapter 9

Results for American-Bermuda-Asian option

Finally we get to the pricing of American Asian options. We use the results above to obtain the best approximate for the American Asian options as possible, without sacrificing computational efficiency. We use 1000 simulations to approximate the American average options on weighted Laguerre basis functions with 2 low discrepancy sequences (e.g Halton and Sobol) as well as with the Pseudo random sequence. The Faure sequence yielded inaccurate results when approximating the American average options and has therefore been excluded from the pricing in this section.

We price an American Asian option with the following attributes. The strike price for this option is $K = 100$, the initial average of the option is denoted by A, the risk-free interest rate is $r = 6\%$, the underlying price is denoted by S, and the initial underlying price is denoted by S_0. The time to maturity for the option is two years and has 100 discretization points per year. The volatility of the underlying is assumed to be 0.2 and the average of the stock price is calculated over three months, prior to the exercise date. We expect that the

calculation of the American Asian option will demand more computational effort due to the calculation of the average share price over a three-month look back period.

Table 9.1: Error Analysis for American Average options

	Halton	Sobol	CG
Max Error	0.0091	0.0111	0.0055
Min Error	1.0737	0.4365	1.3659
Error stdev	0.2973	0.1357	0.4279
% avg est error	3.9478%	1.1535%	4.0010%
Avg CPU time	0.4135	0.4687	0.3265

From table 9.1, the error analysis that was performed confirmed that the Sobol sequence yielded the smallest relative error of approximately 1.15%, the Halton sequence yielded an error of 3.9478%, and the Pseudo random sequence yielded an error of 4%. The Halton and Pseudo random sequence delivered results with relatively high errors under a low number of simulations. The error standard deviation for the above mentioned three sequences was 13.57 cents, 29.73 cents, and 42.79 cents respectively. In terms of computational effort required to run the Asian option algorithm, it took the Quasi random sequences approximately 0.1 seconds longer to execute the algorithm, but with a significant increase in accuracy from the Sobol sequence.

Table 9.2: American average options with Low discrepancy sequences

		Finite diff	Sobol		Halton		CG	
A	S_0	Amer Avg	Amer Avg	(s.e.)	Amer Avg	(s.e.)	Amer Avg	(s.e.)
90	80	0.949	0.9862	0.0369	0.9193	0.0356	0.9545	0.0253
90	90	3.267	3.2792	0.0698	3.1904	0.0678	3.1555	0.0461
90	100	7.889	7.8162	0.1054	8.05	0.1075	7.6766	0.089
90	110	14.538	14.723	0.1346	14.8432	0.1395	14.9063	0.1374
90	120	22.423	22.6436	0.1236	23.0136	0.1436	23.0624	0.1562
100	80	1.108	1.1191	0.04	0.93	0.0358	1.0888	0.0355
100	90	3.71	3.7486	0.0769	3.5646	0.0734	3.503	0.0678
100	100	8.658	8.6243	0.1151	8.5465	0.1133	8.3845	0.1063
100	110	15.717	15.6696	0.1472	15.7079	0.1475	15.3708	0.1402
100	120	23.811	23.8635	0.1534	24.1205	0.1694	23.7964	0.1702
110	80	1.288	1.3199	0.0446	1.2448	0.0431	1.1285	0.0363
110	90	4.136	4.1233	0.0824	4.0495	0.0805	3.9955	0.0745
110	100	9.821	9.8061	0.1294	9.1695	0.1206	9.0448	0.1134
110	110	17.399	16.9625	0.1642	16.3253	0.1551	16.0331	0.1496
110	120	25.453	25.0883	0.180448	25.0004	0.1783	24.2447	0.178

Chapter 10

GPU Optimization of Longstaff and Schwartz

GPU's, or graphical processing units, were primarily designed because of an increasing demand for computers to process and display sophisticated graphics [14]. Graphic calculations have a parallel nature; GPUs were specifically designed with the ability to process these calculations. In many circumstances it has been discovered that it is more beneficial to employ GPU's for image and graphical processing. The most distinguishing factors that separates a Central processing unit or CPU from a GPU is with focus on throughput rather than latency. There are, however, sacrifices when working with GPUs. GPUs generally have slower clock speeds compared to CPUs, but a GPU has an extensive amount of processors. Thus, it make sense that GPUs are excellent at tasks, which are mathematically independent, and can process a large number of parallel instructions. GPUs necessarily do not excel at mathematically complex problems.

10.1 Graphical processing unit specifications

A graphical processing unit reaches an almost perfect balance between speed and power. The central processing unit has a higher clock speed and consumes less power in comparison with a graphical processing unit. It is crucial to understand how the graphical processing unit is built to enable us to program it efficiently. When we look deeper into a GPU, we see that a fundamental unit in the GPU is a CUDA core; a combined set of CUDA cores forms a streaming multiprocessor (SM). The streaming processor is comparable to a central processing units core. The programming language that is used on GPUs is known as CUDA programming. In this project, a Nvidea GeForce GTX 1060 6GB is available for use. This device has 10 streaming multiprocessors, which consists of 128 processors each, with a total of 1280 CUDA cores. A computer that has both a central processing unit and graphical processing unit is called a heterogeneous computer, referring to the CPU as the host and GPU as the device. We also have that if the CPU and GPU resides on the same chip then they share common memory. This setup of CPU and GPU will have low latency as the memory is shared and do not have to be tranfered between host and device. In our case we are working with a discrete device where the GPU device has its own memory and if any computations are done, we will have to transfer the memory from device to host and vice versa.

Figure 10.1: CPU and GPU architectures [1]

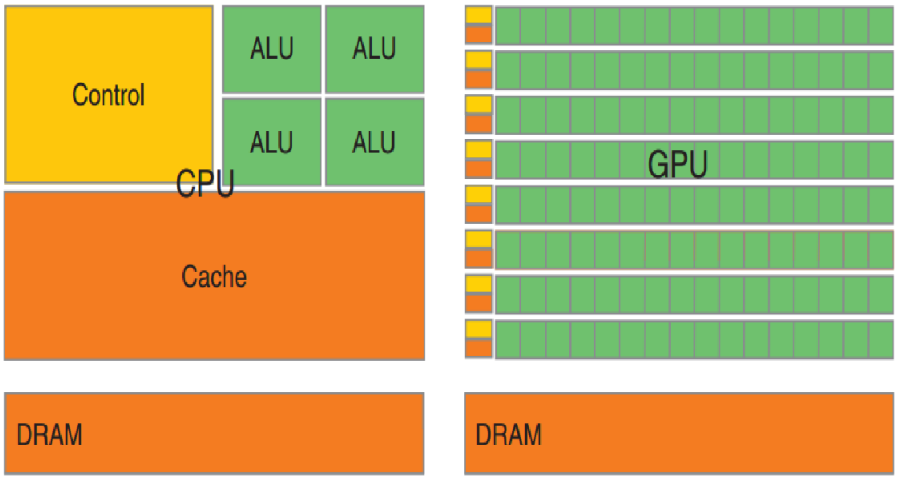


Figure 10.1 illustrates the structural differences between central processing units and graphical processing units. As seen from figure 10.1, the GPU is supplied with multiple weaker cores compared to the strong arithmetic logical units (ALUs) of the central processing unit. These are massive high performance cores, but there are only a few of them. Smaller versions of the control unit and shared caches are shared between these streaming processors on the GPU. Newer generations of central processing units have multiple cores and each core has a hierarchy of caches and a comprehensive control unit that is enabled to handle speculation and out-of-order execution of instructions. The streaming multiprocessor consists of 128 analogous processors, which share a small amount of local memory. This local memory consists out of an instruction cache, L1 data cache, and shared memory. Streaming multiprocessors work together on a single task at any given time. The streaming multiprocessors use shared memory to process a single task at a time. This is computationally cheap as shared memory is characterized to have very small latency. However, the shared memory is of limited in

size. Parallel computing on the GPU is accomplished by launching multiple number of threads that execute independent [14] parts of the code. Threads, however, can be turned into blocks and then these blocks are usually organised in a grid. Then one of these blocks (consisting out of threads) gets automatically assigned to a single streaming multiprocessor. Not all these threads, in a certain block, are executed concurrently. Threads of a certain block that are executed concurrently are called warps and warps usually consist of 32 threads. From Shchekina [14], if an instruction follows a conditional statement, then the threads that shall not enter the statement will make the calculation nonetheless. However, this calculation will not yield a result.

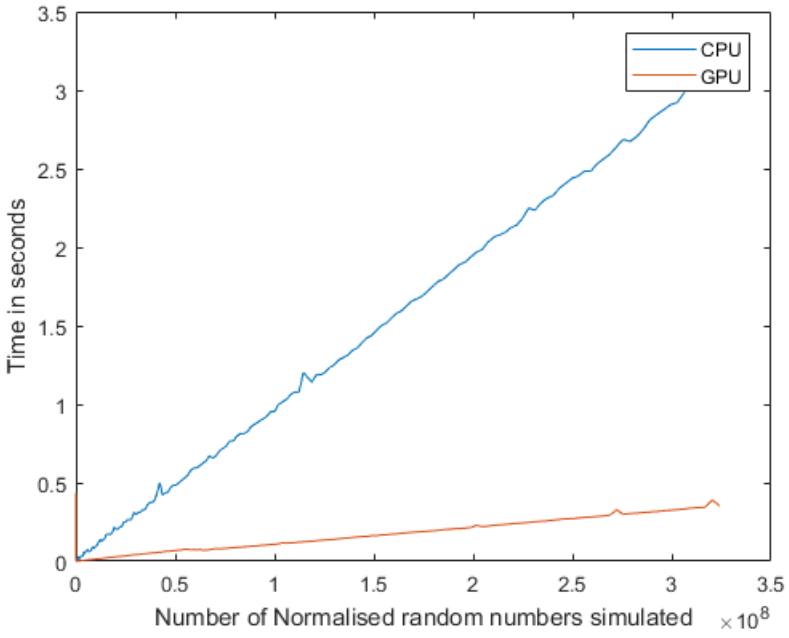
Different warps are handled by the GPU thread scheduler and multiple active warps in the SM would mean that the GPU is using its resources efficiently. Central processing units and graphical processing units handle instructions in different ways. The Central processing unit uses integrated tools, like the branch predictor that is included in its control unit and enables the central processing unit to minimize latency. However, a graphical processing unit will switch from the one thread to another if the one takes either too long or if it is waiting for memory. This is a process that the GPU implements to reduce latency. Hence, the GPU will produce a good throughput if there are enough threads to pile up.

10.2 Negative aspects of using a GPU

A heterogeneous computer is the most well-known computer available to consumers. However, as specified in the previous section a heterogeneous computer consists out of a host and device, CPU and GPU respectively. This type of configuration creates a bottleneck for data, especially when data needs to be transferred between the GPU and CPU. The graphical processing unit generally has better bandwidth than a central processing unit. But the

transfer between the host and device is extremely slow.

Figure 10.2: Generating normalised random numbers on CPU and GPU



In figure 10.2, we generated normalized random numbers on both the GPU and CPU concurrently and recorded the time in seconds that it took to generate these random numbers. We observe that at 1 million simulations the CPU takes 0.0156 seconds and the GPU takes around 0.002 seconds. This makes the GPU around 7.8 times faster than the CPU. If we move on and look at 100 million simulations, the CPU takes 0.9844 seconds and the GPU takes around 0.1081. This means that the GPU completed its simulations approximately 9.1 times faster. We need to keep in mind that these times are restricted to the current computational setup. We used a Intel i7-7700K central processing unit versus a Nvidia GeForce GTX 6600 6GB graphics card.

10.3 Matlab: GPU optimized American option

We consider a similar example as seen in section 9. We run the Longstaff and Schwartz algorithm and observe the time it takes for the algorithm to execute on both the central processing unit and the graphical processing unit. We expect a marginal improvement of the performance of the algorithm as it is more efficient to generate random numbers on the GPU. But once these numbers are generated the data needs to be transferred to the central processing unit for the rest of the algorithm to execute.

Table 10.1: Longstaff and Schwartz approximations: CPU vs GPU

S0	T	sigma	N	Time CPU	CPU approx	TimeGPU	GPU approx
36	1	0.2	100,000	4.256	4.4174	3.9844	4.401
36	1	0.2	250,000	9.3594	4.4104	9.3591	4.4162
36	1	0.2	500,000	17.4531	4.4105	17.125	4.4118
36	1	0.2	750,000	26.0938	4.4114	25.2188	4.4094
36	1	0.2	1,000,000	34	4.412	33.2813	4.4141
36	1	0.2	10,000,000	359.5313	4.415	359.125	4.415

In table 10.1, we observe that for 100 000 simulations, which is denoted by N, that there is a 0.2716 difference in the execution time of the algorithm. The GPU just outperforms the CPU. The graphical processing unit outperforms the central processing unit on each of the increments of simulation, but only by a small margin. This is likely caused by the bottleneck in heterogeneous computers, where data needs to be transferred between the graphical processing unit and the central processing unit. Matlab is a GPU enabled platform and it enables a person to send certain instructions to the GPU of a computer. However, the data sent to the GPU needs to be retrieved and implemented with the algorithm on the CPU concurrently.
This in comparison with table 10.2, where a significant difference in sim-

ulation time is observed, means that the heterogeneous computer creates a bottleneck when the data is transferred between host and device. The problem can be addressed by writing pure CUDA code and implementing the whole Longstaff and Schwatrz algorithm on the graphical processing unit.

Table 10.2: Random number simulation on both CPU and GPU

N	CPU time	GPU time
1.00E+06	0.015625	0.002095
1.21E+06	0.015625	0.002316
1.44E+06	0.015625	0.002986
1.69E+06	0.015625	0.003287
1.96E+06	0.03125	0.00328
2.25E+06	0.015625	0.003735
...	...	...
3.10E+08	3.125	0.38516
3.13E+08	3.140625	0.383419
3.17E+08	3.203125	0.402779
3.20E+08	3.203125	0.389849
3.24E+08	3.265625	0.348078

[N represents the number of simulations that are either performed by the central processing unit or the graphical processing unit. The CPU time and GPU time, repsectively, give the time it took in secodes to complete the specific number of simulations.]

Chapter 11

Implied and Local volatility surfaces for Asian options

As we discussed previously, there exists no closed form solution for an American arithmetical Asian option. Thus implementing local volatility pricing methodology, which is used to price exotic options and express them in terms of vanilla options, will increase the efficiency with which these options are priced. According to Dr A Kotze [23] most exotic options that are listed on the JSE are estimated using local volatility models, which need a local volatility surface.

A breakthrough was made in 1994 by Bruno Dupire [19], Emanuel Derman and Iraj Kani [22]. They noted that under the risk neutrality assumption that, [18], there was a unique diffusion process that was consistent with these distributions. The corresponding state dependent diffusion coefficient $\sigma_L(S,t)$, which was consistent with European option prices, is better known as the local volatility function.

The following [18] explains the state of mind of Dupire, Derman, and Kani best, "It is unlikely that Dupire, Derman, and Kani ever thought of local volatility as representing a model of how volatilities actually evolve. Rather,it is likely that they thought of local volatilities as representing some kind of

average over all possible instantaneous volatilities in a stochastic volatility world. Local volatility models do not therefore really represent a separate class of models; the idea is more to make a simplifying assumption that allows practitioners to price exotic options consistently with the known prices of vanilla options."
The main difference is that Dupire published a model in continuous time theory, while Derman and Kani investigated a discrete binomial tree version. Using Stochastic volatility models has proven to be very useful [18], because these models explain why options with different strikes and different expiration will yield different implied volatilities under the assumption of the Black-Scholes model. Illustrating these implied volatilities in a graphical manner will yield a volatility smile.

11.1 Derivation of Dupire's equation

For the Proof of Dupire's equation, we will follow the guidelines of J. Gatheral [18]. A underlying stock S_0, at a given maturity T, will yield a collection of undiscounted option prices $\{C(S_0, K, T)\}$ for different strike prices K. This will yield the risk neutral probability density function φ of the final spot price S_T through the following relationship.

$$C(S_0, K, T) = \int_K^\infty dS_T \varphi(S_T, T; S_0)(S_T - K) \qquad \text{(eq:11.1.1)}$$

Suppose then that the stock price diffuses with the risk neutral drift μ_t and local volatility $\sigma(S, t)$ according to the following stochastic differential equation that evolves log normally

$$dS = \mu_t dt + \sigma(S_t, t) dZ \qquad \text{(eq:11.1.2)}$$

The application of Itô with risk neutrality provides us with the partial differential equation for functions of the stock price. This partial differential equa-

tion is just a generalization of the Black-Scholes partial differential equation. Let us then assume that the probability density is given by $\varphi(K,T;S_0) = \frac{\partial^2 C}{\partial K^2}$ and this must satisfy the Fokker-Planck equation. This gives us the following partial differential equation for strike price K and option price C:

$$\frac{\partial C}{\partial T} = \frac{\sigma^2 K^2}{2}\frac{\partial^2 C}{\partial K^2} + (r_f - D)(C - K\frac{\partial C}{\partial K}) \tag{eq:11.1.3}$$

where r_f is the corresponding risk-free rate, D is the dividend rate, and C is short for the corresponding option price.
Now the pseudo-probability density φ will evolve according to the Fokker-Plack equation.

$$\frac{1}{2}\frac{\partial^2}{\partial S_T^2}(\sigma^2 S_T^2 \varphi) - S\frac{\partial}{\partial S_T}(\mu S_T \varphi) = \frac{\partial \varphi}{\partial T} \tag{eq:11.1.4}$$

Now if we differentiate equation 11.1.1 with respect to K, we obtain

$$\frac{\partial C}{\partial K} = -\int_K^\infty dS_T \varphi(S_T, T; S_0) \tag{eq:11.1.5}$$

$$\frac{\partial^2 C}{\partial K^2} = \varphi(K, T; S_0) \tag{eq:11.1.6}$$

and differentiating equation 11.1.1, with respect to time or T, yields

$$\begin{aligned}
\frac{\partial C}{\partial T} &= \int_K^\infty dS_T [\frac{\partial}{\partial T}\varphi(S_T, T; S_0)](S_T - K) \\
&= \int_K^\infty dS_T [\frac{1}{2}\frac{\partial^2}{\partial S_T^2}(\sigma^2 S_T^2 \varphi) - \frac{\partial}{\partial S_T}(\mu S_T \varphi)](S_T - K) \\
&= \int_K^\infty dS_T [\frac{1}{2}\frac{\partial^2}{\partial S_T^2}(\sigma^2 S_T^2 \varphi)(S_T - K)] - \int_K^\infty dS_T [\frac{\partial}{\partial S_T}(\mu S_T \varphi)](S_T - K)]
\end{aligned} \tag{eq:11.1.7}$$

Now we do integration by parts. We do integration by parts twice on the first integral and once on the second integral. Hence we obtain,

$$\begin{aligned}\frac{\partial C}{\partial T} &= \frac{1}{2}\sigma^2 K^2 \varphi + \int_K^\infty dS_T \mu S_T \varphi \\ &= \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2} + \mu(T)(-K\frac{\partial C}{\partial K}) \\ &= \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2} + (r_f - D)(-K\frac{\partial C}{\partial K})\end{aligned} \tag{eq:11.1.8}$$

This then yields the Dupire equation. We define the forward price of a stock at time T as

$$F_T = S_0 \exp \int_0^T \mu_t dt$$

Now we attempt to express the option price as a function of the forward price. Hence we would be able to obtain the same expression without the drift term. In this case,

$$\frac{\partial C}{\partial T} = \frac{1}{2}\sigma^2 K^2 \frac{\partial^2 C}{\partial K^2} \tag{eq:11.1.9}$$

Hence, by inverting the equation that we found in equation 11.1.9 and we solve for σ^2, we obtain the following expression.

$$\sigma^2_{local} = \frac{2\frac{\partial C}{\partial T}}{K^2 \frac{\partial^2 C}{\partial K^2}} \tag{eq:11.1.10}$$

We can use equation 11.1.10, which can be calculated from known European option prices. Therefore, the local volatilities can be expressed, as seen in equation 11.1.10, given the set of option prices for all strikes and expirations. Hence, equation 11.1.10 serves as a definition for local volatility, regardless of the process that drives the volatility.

11.2 Local volatility expressed in terms of implied volatility

We assume that the market prices of options will be quoted in terms of the Black-Scholes implied volatility [18], denoted by $\sigma_{BS}(K, T; S_0)$. Hence we obtain that,

$$C(S_0, K, T) = C_{BS}(S_0, K, \sigma_{BS}(K, T; S_0), T)$$

If we use and manipulate the pricing formula of Black-Scholes, we obtain the following (from theorem 3):

$$\begin{aligned} C_{BS}(S_T, K, \sigma_{BS}(K, T; S_0)) &= S_T N(d_1) - K N(d_2) \\ &= F_T[N(d_1) - e^y N(d_2)] \end{aligned} \qquad \text{(eq:11.2.1)}$$

where $F_T = S_0 \exp \int_0^T \mu_t dt$ is the forward price of a stock and $y = \log(\frac{K}{F_T})$. Specifying the terms as indicated here will be more convenient to work with 2 dimensionless variables. We furthermore define the Black-Scholes total implied variance and the log-strike as,

$$\omega(S_0, K, T) = \sigma_{BS}(S_0, K, T)T$$

$$y = \log\left(\frac{K}{F_T}\right)$$

So, we are able to further simplify equation 11.2.1, as follows:

$$\begin{aligned} C_{BS}(S_T, K, \sigma_{BS}(K, T; S_0)) &= F_T[N(d_1) - e^y N(d_2)] \\ &= F_T[N(-\frac{y}{\sqrt{\omega}} + \frac{\sqrt{\omega}}{2}) - e^y N(-\frac{y}{\sqrt{\omega}} - \frac{\sqrt{\omega}}{2}) \end{aligned} \qquad \text{(eq:11.2.2)}$$

We are then able to find Dupire's equation as in equation 11.1.3,

$$\frac{\partial C}{\partial T} = \frac{v_L}{2}(\frac{\partial^2 C}{\partial y^2} - \frac{\partial C}{\partial y}) + \mu(T)C \qquad \text{(eq:11.2.3)}$$

where v_L now denotes the local variance $v_L = \sigma^2(S_0, K, T)$. Thus by taking derivatives with respect to the Black Scholes formula $C_B S$, we are able to obtain,

$$\begin{aligned} \frac{\partial^2 C_{BS}}{\partial \omega^2} &= (-\frac{1}{8} - \frac{1}{2\omega} + \frac{y^2}{2\omega^2})\frac{\partial C_{BS}}{\partial \omega} \\ \frac{\partial^2 C_{BS}}{\partial y \partial \omega} &= (\frac{1}{2} - \frac{y}{\omega})\frac{\partial C_B S}{\partial \omega} \\ \frac{\partial^2 C_{BS}}{\partial y^2} - \frac{\partial C_{BS}}{\partial y} &= 2\frac{\partial C_{BS}}{\partial \omega} \end{aligned} \qquad \text{(eq:11.2.4)}$$

We now substitute these terms back into the Dupire equation 11.2.3 and transform the equation in terms of implied variance,

$$\begin{aligned} \frac{\partial C}{\partial y} &= \frac{\partial C_{BS}}{\partial y} + \frac{\partial C_{BS} \partial \omega}{\partial \omega \partial y} \\ \frac{\partial^2 C}{\partial y^2} &= \frac{\partial^2 C_{BS}}{\partial y^2} + 2\frac{\partial^2 C_{BS} \partial \omega}{\partial y \partial \omega \partial y} + \frac{\partial^2 C_{BS}}{\partial \omega^2}(\frac{\partial \omega}{\partial y})^2 + \frac{\partial C_{BS}}{\partial \omega}\frac{\partial^2 \omega}{\partial y^2} \\ \frac{\partial C}{\partial T} &= \frac{\partial C_{BS}}{\partial T} + \frac{\partial C_{BS}}{\partial \omega}\frac{\partial \omega}{\partial T} \\ &= \frac{\partial C_{BS}}{\partial \omega}\frac{\partial \omega}{\partial T} + \mu(T)C_{BS} \end{aligned} \qquad \text{(eq:11.2.5)}$$

If we combine and subtract equation 11.2.5 from equation 11.2.3 we are able to cancel out the μ drift term for the equations. We also substitute the

expressions obtained from equation 11.2.5. Hence, the equation becomes

$$\begin{aligned}
0 &= -\frac{\partial C_{BS}}{\partial \omega}\frac{\partial \omega}{\partial T} + \frac{v_L}{2}(\frac{\partial^2 C}{\partial y^2} - \frac{\partial C}{\partial y}) \\
\frac{\partial C_{BS}}{\partial \omega}\frac{\partial \omega}{\partial T} &= \frac{v_L}{2}[-\frac{\partial C_{BS}}{\partial y} - \frac{\partial C_{BS}\partial \omega}{\partial \omega \partial y} + \frac{\partial^2 C_{BS}}{\partial y^2} + 2\frac{\partial^2 C_{BS}\partial \omega}{\partial y \partial \omega \partial y} + \frac{\partial^2 C_{BS}}{\partial \omega^2}(\frac{\partial \omega}{\partial y})^2 \\
&\quad + \frac{\partial C_{BS}}{\partial \omega}\frac{\partial^2 \omega}{\partial y^2}] \\
&= \frac{v_L}{2}\frac{\partial C_{BS}}{\partial \omega}[2 - \frac{\partial \omega}{\partial y} + 2(\frac{1}{2} - \frac{y}{\omega})\frac{\partial \omega}{\partial y} + (-\frac{1}{8} - \frac{1}{2\omega} + \frac{y^2}{2\omega^2})(\frac{\partial \omega}{\partial y})^2 \\
&\quad + \frac{\partial^2 \omega}{\partial y^2}]
\end{aligned} \tag{eq:11.2.6}$$

We now cancel out the $\frac{\partial C_{BS}}{\partial \omega}$ term and simplify the equation to the following:

$$\frac{\partial \omega}{\partial T} = v_L\left(1 - \frac{y}{\omega}\frac{\partial \omega}{\partial y} + \frac{1}{4}\left(- \frac{1}{4} - \frac{1}{\omega} + \frac{y^2}{\omega^2}\right)\left(\frac{\partial \omega}{\partial y}\right)^2 + \frac{1}{2}\frac{\partial^2 \omega}{\partial y^2}\right)$$

Solving for our local volatility estimator yields

$$v_L = \frac{\frac{\partial \omega}{\partial T}}{\left(1 - \frac{y}{\omega}\frac{\partial \omega}{\partial y} + \frac{1}{4}\left(- \frac{1}{4} - \frac{1}{\omega} + \frac{y^2}{\omega^2}\right)\left(\frac{\partial \omega}{\partial y}\right)^2 + \frac{1}{2}\frac{\partial^2 \omega}{\partial y^2}\right)}$$

This v_L parameter will estimate the value of the local volatility of a certain model and will represent it in the Black Scholes pricing framework.

11.3 American Asian option Implied and Local volatility surfaces

Local volatility surfaces are fairly common in the pricing of exotic options. The industry uses these surfaces to price options, which are complex to price, quickly and efficiently. Reference prices can efficiently be obtained through

the implementation of these surfaces. In this section, we investigate an example by setting up the implied and local volatility surfaces for American Asian options.
The basic algorithm that we implement to enable us to find the local volatilities can be described as follows.

1. We simulate the prices for our exotic options. These prices consist of a matrix of prices, at different maturities and different strike prices.

2. We then calculate the corresponding implied volatility estimates with the Black-Scholes formula for different maturity and different strike price pairs.

3. We use Dupire's equation to obtain local volatility estimates from the implied volatility estimates. We also perform the necessary numerical differentiation according to Dupire's equation.

4. Lastly, we linearly interpolate between the local volatility estimates. This enables us to plot the respective surface.

11.3.1 Example: American Asian call option: Implied and Local volatility surfaces

For illustration purposes we propose the following problem. We will be pricing an American Asian call option with the Longstaff and Schwartz model. We start pricing this option at an initial time of 90 days and end at 525 days, with 15-day increments. For different strike prices we start to price at a strike price of 40 and end at 74 and move in increments of 2. Our risk-free rate and initial underlying price is 6% and 50, respectively. We also work under the assumption that there are 252 business days in a year.

Table 11.1: Example assumptions

S_0	50
σ	20%
r	6%
Time base	252
K	40-74
Time start	90 days
Time end	525 days
time increment	15 days
Exercised per year	50

Figure 11.1: Plot of Implied volatility of American Asian call

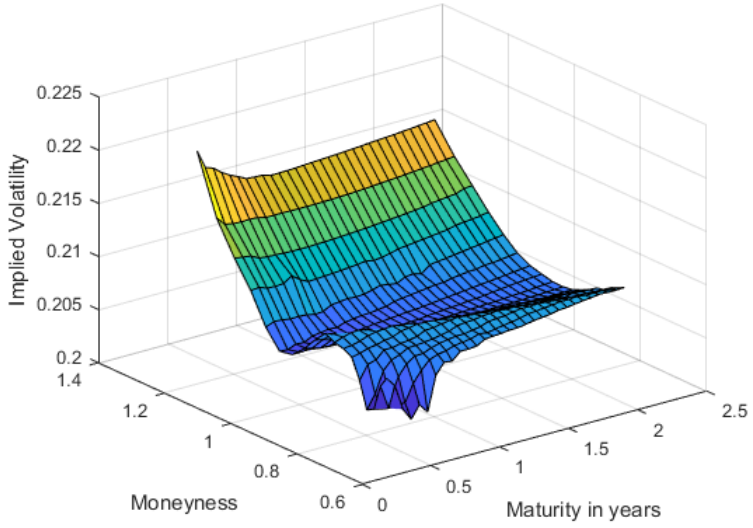
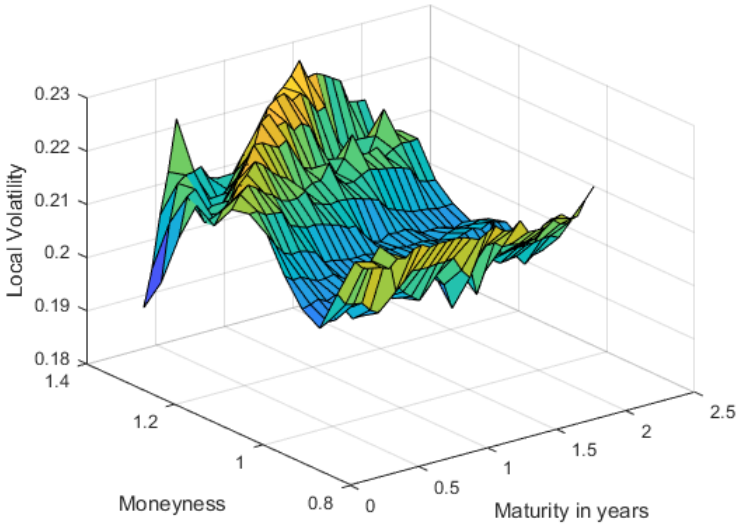


Table 11.2: Simulated prices with the Longstaff and Schwartz algorithm

		90	105	120	135	150	165	...	510	525
Strike (K)	40	11.01674	11.11076	11.21496	11.32491	11.41368	11.52072	...	13.90595	14.01013
	42	9.152381	9.277942	9.398501	9.507196	9.625552	9.742451	...	12.32132	12.42368
	44	7.253717	7.386304	7.521604	7.657134	7.792707	7.92791	...	10.78793	10.90474
	46	5.28E+00	5.448857	5.603429	5.759075	5.914247	6.055695	...	9.065651	9.185394
	48	3.655111	3.847565	4.03488	4.206236	4.378461	4.544344	...	7.778591	7.904379
	50	2.344209	2.559037	2.753663	2.932275	3.117916	3.290823	...	6.571758	6.69724
	52	1.573276	1.747914	1.917454	2.062113	2.210597	2.332075	...	5.516085	5.653886
	54	1.037472	1.194009	1.335339	1.508285	1.655985	1.78551	...	4.588224	4.714035
	56	0.639611	7.85E-01	0.9262	1.062035	1.190756	1.311836	...	3.784867	3.888785
	58	0.377038	0.483569	0.59367	0.720242	0.832254	0.954953	...	3.159323	3.234658
	60	0.210375	0.295246	0.382128	0.478064	0.556508	0.664749	...	2.723003	2.789182
	62	0.108362	0.163151	0.233446	0.309436	0.383569	0.4592	...	2.104816	2.22463
	64	0.058882	0.09066	0.132723	0.180758	0.261989	0.301319	...	1.859399	1.93031
	66	0.030097	0.053365	0.075538	0.104574	0.132776	0.150554	...	1.578348	1.633488
	68	0.01455	0.026294	0.04015	0.058687	0.084833	0.1114	...	1.335082	1.424537
	70	0.006359	0.015296	0.02602	0.040406	0.052876	0.074195	...	1.178605	1.270406
	72	0.003223	0.00718	0.013743	0.025644	0.037218	0.048258	...	1.052714	1.079242
	74	0.001738	0.003792	0.007584	0.014596	0.023718	0.035385	...	0.840147	0.911611
	76	0.000865	0.002207	0.0043	0.0082	0.014232	0.022199	...	0.718455	0.789409
	78	0.00034	0.001218	0.002498	0.004735	0.008657	0.014773	...	0.625341	0.660867

Figure 11.2: Plot of Local volatility of American Asian call



We observe from figure 11.2 that Dupire's local volatility function, in terms of the implied volatility, will lead to a practical and usable surface to calculate the exotic options reference price quickly. However, we observe some instabilities in the local volatility graph where the prices are far in and out of the money. We also see that the local volatility surface is not smooth due the the numerical differentiation estimates that we calculated for the Dupire's equation. Both the implied and local volatility surfaces show more or less the same shape. We observe from the local volatility surface that the volatility estimates are generally higher than what we observed from the implied volatility surface.

11.3.2 American Asian put option: Implied and Local volatility surfaces

We will price an American Asian put option using the least squares Monte Carlo algorithm. We start pricing this option at an initial time of 90 days and end at 300 days, with 15 day increments. For different strike prices we start to price at a strike price of 56 and end at 82 and move in increments of 2.We consider the following table as a quick summary of the example that will be considered.

Table 11.3: Example assumptions

S_0	70
r	6%
Time base(days in a year)	252
K	56-82
Time start	90 days
Time end	300 days
time increment	15 days
σ	0.2

Table 11.4: Prices for American Asian put option

		Time in days to matrurity							
		90	105	120	135	150	165	285	300
Strike (K)	56	0.052558	0.079488	0.109384	0.141422	0.175406	0.210555	0.491257	0.524012
	58	0.115644	0.160437	0.207995	0.257238	0.306216	0.355184	0.712946	0.752004
	60	0.232747	0.301356	0.370576	0.438528	0.504597	0.568716	1.001231	1.045519
	62	0.431053	0.527112	0.619391	0.706685	0.78902	0.867046	1.355519	1.40378
	64	0.730646	0.846415	0.962194	1.069418	1.169661	1.259055	1.772792	1.823178
	66	1.163615	1.300706	1.433528	1.548347	1.653322	1.749819	2.294817	2.345422
	68	1.770661	1.920297	2.053193	2.175161	2.28037	2.380999	2.912931	2.960986
	70	2.560516	2.706914	2.838626	2.953534	3.05971	3.14407	3.640263	3.682931
	72	3.536415	3.664731	3.777734	3.874924	3.974033	4.057713	4.463057	4.497617
	74	4.688249	4.78822	4.875019	4.949326	5.021934	5.087197	5.383278	5.406612
	76	6.011924	6.0697	6.116348	6.16548	6.202136	6.242107	6.399883	6.409681
	78	7.484198	7.495697	7.502052	7.50857	7.515068	7.521182	7.50289	7.496734
	80	9.081564	9.044147	9.001641	8.966482	8.939919	8.911588	8.69658	8.671536
	82	10.7721	10.68542	10.60075	10.52688	10.46386	10.40016	9.970536	9.925383
	84	12.54514	12.40699	12.2774	12.16328	12.06508	11.96959	11.32324	11.25614
	86	14.38107	14.1927	14.02121	13.86884	13.73359	13.5982	12.74034	12.6566
	88	16.25214	16.02582	15.81819	15.63056	15.4604	15.29254	14.22567	14.11938

Figure 11.3: Plot of Implied volatility of American Asian put

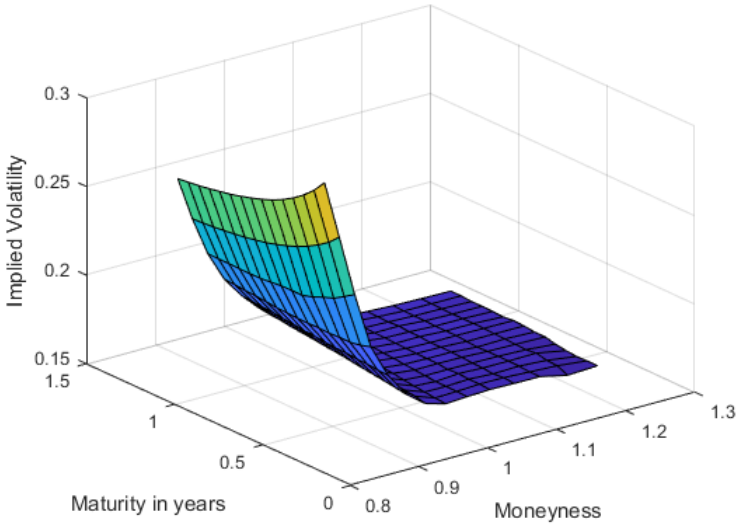
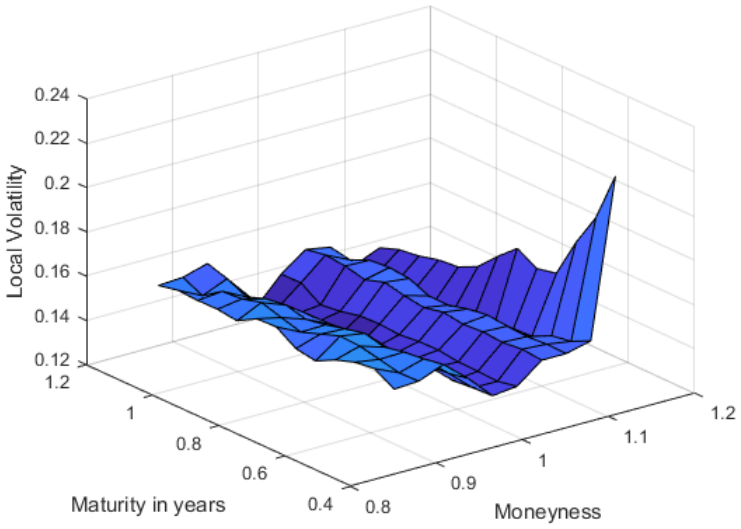


Figure 11.4: Plot of local volatility of American Asian put



We observe from figure 11.4 that the Dupire's local volatility function, in terms of the implied volatility, will lead to a stable approximation of the local volatilities. However once again, we observe that the local volatility surface is not smooth due the the numerical differentiation estimates that we calculated for Dupire's equation. We also observe that the implied and local volatility surfaces does not have the same general shape. For the implementation of this local volatility surface, we had to carefully consider the choice of both strike price range and time to maturity concurrently. It was observed that both the first and second numerical derivatives, with respect to moneyness, become very small negative approximations and then force the local volatility approximate to become a complex approximation. Therefore, take caution in selecting both strike price and time to maturity ranges when setting up a local volatility surface.

Chapter 12

Conclusion

In conclusion, we found that it is possible to obtain a good approximation for American Asian options under a low number of simulations, implementing low discrepancy sequences and variance reduction techniques. The least squares Monte Carlo algorithm needs around 15 seconds to execute under 100 000 simulations as proposed by Longstaff and Schwartz. On the other hand, we showed that it is possible to obtain an approximation with good accuracy implementing low discrepancy sequences. With 1000 simulations it takes around 0.5 seconds to execute the algorithm and obtain an approximation.
The Sobol sequence proved to be very efficient in the pricing of the American Asian options. It yielded the smallest relative error, irrespective of the choice of basis function and under a relatively low number of simulations, and when implementing antithetic variates.
We were also able to confirm that the least squares method is robust to the choice of basis functions, as Longstaff and Schwartz confirmed in their article. There is minimal deviation from the option price with the implementation of different basis functions.
The implementation of low discrepancy sequences with the least squares method is an efficient way of obtaining relatively good accuracy in the approximation of American option prices as well as path dependent options,

irrespective of the choice of basis functions.

The GPU optimization of the generation of random numbers were exponentially excelled via the graphical processing unit. However, the overall least squares Monte Carlo algorithm was marginally optimized with the "device" or graphical processing unit. We confirmed that there is a bottleneck between the host and device when processed data is transferred from the one to the other. It is possible to circumvent this bottleneck in two possible ways. Firstly, a heterogeneous computer was used to optimize the algorithm. If the computer setup is changed, so that there is no necessity to transfer data between host and device, it will lead to a tremendous speedup of the algorithm. Secondly, Matlab was used to program the algorithm; it requires the data to be transferred between the device and host. If the CUDA code is written, which is a dedicated coding language for modern graphical processing units, there will be no need for the data to be transferred and the algorithm will execute solely on the graphics processing unit.

Some exotic options do not have any analytical solution for pricing. They are usually calculated with simulation or numerical methods, which is computationally very inefficient. The construction of the local volatility surface of the Asian options will enable us to calculate the respective Asian options price rapidly. This amplifies the practical usefulness of local volatility surfaces for Asian options. However, careful attention should be applied when choosing the ranges of strike prices and times to maturity.

Bibliography

[1] David B Kirk and W Hwu Wen-Mei. *Programming massively parallel processors: a hands-on approach*. Morgan kaufmann, 2016.

[2] J.Hull and A. White. Efficient procedures for valuing european and american path dependent options. *Journal of Derivatives*, 1:21–31, 1993.

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

[4] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*, volume 706. John Wiley & Sons, 2013.

[5] Friedrich Hubalek and Carlo Sgarra. On the explicit evaluation of the geometric asian options in stochastic volatility models with jumps. *Journal of Computational and Applied Mathematics*, 235(11):3355–3365, 2011.

[6] Yuyun Guna Winarti, Lienda Noviyanti, and Gatot R Setyanto. The european style arithmetic asian option pricing with stochastic interest rate based on black scholes model. In *AIP Conference Proceedings*, volume 1827, page 020001. AIP Publishing, 2017.

[7] Junkee Jeon, Ji-Hun Yoon, and Myungjoo Kang. Valuing vulnerable geometric asian options. *Computers & Mathematics with Applications*, 71(2):676–691, 2016.

[8] Erik Wiklund. Asian option pricing and volatility, 2012.

[9] Investopedia Staff. Asian Option, November 2003.

[10] FokkerPlanck equation, April 2018. Page Version ID: 836231634.

[11] Timothy Klassen. Simple, fast and flexible pricing of asian options. 2000.

[12] Prasad Chalasani, Somesh Jha, Feyzullah Egriboyun, and Ashok Varikooty. A refined binomial lattice for pricing american asian options. *Review of Derivatives Research*, 3(1):85–105, 1999.

[13] Jérôme Barraquand and Thierry Pudet. Pricing of american path-dependent contingent claims. *Mathematical Finance*, 6(1):17–51, 1996.

[14] Anna Shchekina. Asian option pricing using graphics processing units, 2017.

[15] Massimo Costabile, Ivar Massabó, and Emilio Russo. An adjusted binomial model for pricing asian options. *Review of Quantitative Finance and Accounting*, 27(3):285–296, 2006.

[16] Neliswa B Dyakopu. *Discrete time methods of pricing Asian options*. PhD thesis, University of Western Cape, 2014.

[17] Fischer Black and Myron Scholes. The pricing of options and other corporate liabilities. *Journal of Political Economy*, 81:673, 1973.

[18] Jim Gatheral. *The volatility surface: a practitioner's guide*, volume 357. John Wiley & Sons, 2011.

[19] Bruno Dupire. Pricing and hedging with smiles. *Mathematics of derivative securities*, 1(1):103–111, 1997.

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

[21] Erik Wiklund. Asian option pricing and volatility, 2012.

[22] Emanuel Derman and Iraj Kani. Riding on a smile. *Risk*, 7(2):32–39, 1994.

[23] Antonie Kotzé, Rudolf Oosthuizen, and Edson Pindza. Implied and local volatility surfaces for south african index and foreign exchange options. *Journal of Risk and Financial Management*, 8(1):43–82, 2015.

[24] GH Meisters. Lebesgue measure on the real line. n. i. *University of Nebraska, Lincoln*, 1997.

[25] Elaine B. Barker and John M. Kelsey. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Technical Report NIST SP 800-90Ar1, National Institute of Standards and Technology, June 2015. DOI: 10.6028/NIST.SP.800-90Ar1.

[26] Generating Quasi-Random Numbers - MATLAB & Simulink.

[27] Moshe Arye Milevsky and Steven E Posner. Asian options, the sum of lognormals, and the reciprocal gamma distribution. *Journal of financial and quantitative analysis*, 33(3):409–422, 1998.

[28] Paul Glasserman, Philip Heidelberger, and Perwez Shahabuddin. Asymptotically optimal importance sampling and stratification for pricing path-dependent options. *Mathematical finance*, 9(2):117–152, 1999.

[29] Derrick H Lehmer. Mathematical methods in large-scale computing units. *Ann. Comput. Lab. Harvard Univ.*, 26:141–146, 1951.

[30] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.

[31] R. Kufakunesu. Lecture notes wtw831, August 2016.

[32] Karl Sigman. Lecture notes, August 2007.

[33] J. Goodman. Simple sampling of gaissians, August 26 2005.

[34] John Von Neumann. 13. various techniques used in connection with random digits. *Appl. Math Ser*, 12:36–38, 1951.

[35] Phelim Boyle, Mark Broadie, and Paul Glasserman. Monte carlo methods for security pricing. *Journal of economic dynamics and control*, 21(8):1267–1321, 1997.

[36] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

[37] J. Hull. *Options, Futures, and Other Derivatives*. Options, Futures, and Other Derivatives. Prentice Hall, 2012.

[38] EVAN Turner. The black-scholes model and extensions. 2010.

[39] Francis A. Longstaff and Eduardo S. Schwartz. Valuing american options by simulation: A simple least squares approach. *The Society for Financial Studies*, 14:113–147, 2001.

[40] A.G.Z Kemna and A.C.F Vorst. A pricing method for options based on average asset values. *Journal of Banking and Finance*, 14:113–129, 1990.

[41] X. Wang. Randomized halton sequences. *Mathematical and computer modelling*, 32:887–899, 2000.

[42] Henri Faure. Discrpance de suites associes un systme de numration. *Acta Arithmetica*, 41:337–351, 1982.

[43] Edmond Levy. Pricing european average rate currency options. *Journal of International Money and Finance*, 11:474–491, 1992.

[44] Andrew Carverhill and Les J Clewlow. Flexible convolution. *Risk*, 3:25–29, 1990.

[45] Stuart M. Turnbull and Lee MacDonald Wakeman. A quick algorithm for pricing european average options. *Cambridge University Press*, 26:377–389, 1991.

[46] PricewaterhouseCoopers. 2017. Sas mining industry sees steep decline in financial performance impacted by a slump in commodity prices and increased cost pressures @ONLINE, January 2017.

[47] Investopia. Asian option @ONLINE, 2017.

[48] Moshe Arye Milevsky and Steven E Posner. Asian options, the sum of lognormals, and the reciprocal gamma distribution. *Journal of financial and quantitative analysis*, 33(3):409–422, 1998.

[49] Low-discrepancy sequence, October 2017. Page Version ID: 806063794.

[50] I.M Sobol. Distribution of points in a cube and approximate evaluation of integrals. *U.S.S.R Comput. Maths. Math. Phys.*, 7:86–112, 1967.

[51] J.Crank and E. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6:207–226, 1996.

[52] Stephen Joe and Frances Y Kuo. Remark on algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 29(1):49–57, 2003.

[53] Daniel J Duffy. *Finite Difference Methods in Financial Engineering.* John Wiley 'I&' Sons, Ltd, West Sussex, England, 2006.

[54] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts, 1993.

Appendices

Appendix A: Definitions

Definition 4 (Inner product space)**.** Let E be a complex vector space. A mapping $\langle , \rangle \; ExE \to C$, is called an inner product in E if for any $x, y, z \in E$ the following conditions are satisfied,

1. $\langle x, y\rangle = \langle \bar{y, x}\rangle$

2. $\langle \alpha x + \beta y, z\rangle = \alpha\langle x, z\rangle + \beta\langle y, z\rangle$

3. $\langle x, x\rangle \geq 0$

4. $\langle x, x\rangle = 0$ implies that $x = 0$

Definition 5 (Hilbert space)**.**
A complete inner product space is called a Hilbert space.

Definition 6 (Lebesgue measurable [24])**.**
We define a bounded interval as I with endpoints a and b where $(a < b)$. The length of I is then defined by $l(I) = b - a$ and if I is (a, ∞),$(-\infty, b)$ or $(-\infty, \infty)$, then the length of the interval will be $l(I) = \infty$.
Then given a set E of real numbers, $\mu(E)$ will denote the Lebesgue measure if it is defined. Preferable properties:

1. Extends length: For every interval I, $\mu(I) = l(I)$.

2. **Monotone:** If $A \subset B \subset \mathbb{R}$, then $0 \leq \mu(A) \leq \mu(B) \leq \infty$.

3. **Translation invariant:** For each subset A of $\mathbb{R}$and for each point $x_0 \in \mathbb{R}$ we define $A + x_0 := \{x + x_0 : x \in A\}$. Then $\mu(A + x_0) = \mu(A)$.

4. **Countable additive:** If A and B are disjoint subsets of $\mathbb{R}$, then $\mu(A \cup B) = \mu(A)+\mu(B)$. If $\{A_i\}$ is a sequnce of disjoint sets, then $\mu(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$.

Definition 7 (Fokker-Planck equation [10])**.**
In one spatial dimension and a standard Wiener process W_t. The stochastic differential equation;

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$$

with drift $\mu(X_t, t)$ and diffusion coefficient $D(X_t, t) = \frac{\sigma^2(X_t,t)}{2}$. The the Fokker Planck equation for the probability density $p(x, t)$ of the random variable X_t is

$$\frac{\partial}{\partial t}p(x, t) = -\frac{\partial}{\partial x}[\mu(x, t)p(x, t)] + \frac{\partial^2}{\partial x^2}[D(x, t)p(x, t)]$$

Definition 8 (Integration by parts)**.**

$$u\frac{dv}{dx}dx = uv - \int \frac{du}{dx}vdx$$

Appendix B: Code

Appendix B is attached to this dissertation, as either a CD or USB. It contains the MATLAB codes that were used in this dissertation.