UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

Calculation Aspects of the European Rebalanced Basket Option using Monte Carlo Methods

by

Carel Johannes van der Merwe

Assignment presented at the University of Stellenbosch in partial fulfilment of the requirements for the degree of

Masters of Commerce

Department of Statistics and Actuarial Science
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Prof. W.J. Conradie

December 2010

To my Mother, Father and sisters.

For 25 years of everything.

DECLARATION

I, the undersigned, hereby declare that the work contained in this assignment is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .....

C.J. van der Merwe

Date: .....

ABSTRACT

Life insurance and pension funds offer a wide range of products that are invested in a mix of assets. These portfolios (Π), underlying the products, are rebalanced back to predetermined fixed proportions on a regular basis. This is done by selling the better performing assets and buying the worse performing assets. Life insurance or pension fund contracts can offer the client a minimum payout guarantee on the contract by charging them an extra premium (α). This problem can be changed to that of the pricing of a put option with underlying Π. It forms a liability for the insurance firm, and therefore needs to be managed in terms of risks as well. This can be done by studying the option's sensitivities. In this thesis the premium and sensitivities of this put option are calculated, using different Monte Carlo methods, in order to find the most efficient method.

Using general Monte Carlo methods, a simplistic pricing method is found which is refined by applying mathematical techniques so that the computational time is reduced significantly. After considering Antithetic Variables, Control Variates and Latin Hypercube Sampling as variance reduction techniques, option prices as Control Variates prove to reduce the error of the refined method most efficiently. This is improved by considering different Quasi-Monte Carlo techniques, namely Halton, Faure, normal Sobol' and other randomised Sobol' sequences. Owen and Faure-Tezuke type randomised Sobol' sequences improved the convergence of the estimator the most efficiently. Furthermore, the best methods between Pathwise Derivatives Estimates and Finite Difference Approximations for estimating sensitivities of this option are found.

Therefore by using the refined pricing method with option prices as Control Variates together with Owen and Faure-Tezuke type randomised Sobol' sequences as a Quasi-Monte Carlo method, more efficient methods to price this option (compared to simplistic Monte Carlo methods) are obtained. In addition, more efficient sensitivity estimators are obtained to help manage risks.

OPSOMMING

Lewensversekering en pensioenfondse bied die mark 'n wye reeks produkte wat belê word in 'n mengsel van bates. Hierdie portefeuljes (Π), onderliggend aan die produkte, word op 'n gereelde basis terug herbalanseer volgens voorafbepaalde vaste proporsies. Dit word gedoen deur bates wat beter opbrengste gehad het te verkoop, en bates met swakker opbrengste aan te koop. Lewensversekering- of pensioenfondskontrakte kan 'n kliënt 'n verdere minimum uitbetaling aan die einde van die kontrak waarborg deur 'n ekstra premie (α) op die kontrak te vra. Die probleem kan verander word na die prysing van 'n verkoopopsie met onderliggende bate Π. Hierdie vorm deel van die versekeringsmaatskappy se laste en moet dus ook bestuur word in terme van sy risiko's. Dit kan gedoen word deur die opsie se sensitiwiteite te bestudeer. In hierdie tesis word die premie en sensitiwiteite van die verkoopopsie met behulp van verskillende Monte Carlo metodes bereken, om sodoende die effektiefste metode te vind.

Deur die gebruik van algemene Monte Carlo metodes word 'n simplistiese prysingsmetode, wat verfyn is met behulp van wiskundige tegnieke wat die berekeningstyd wesenlik verminder, gevind. Nadat Antitetiese Veranderlikes, Kontrole Variate en Latynse Hiperkubus Steekproefneming as variansiereduksietegnieke oorweeg is, word gevind dat die verfynde metode se fout die effektiefste verminder met behulp van opsiepryse as Kontrole Variate. Dit word verbeter deur verskillende Quasi-Monte Carlo tegnieke, naamlik Halton, Faure, normale Sobol' en ander verewekansigde Sobol' reekse, te vergelyk. Die Owen en Faure-Tezuke tipe verewekansigde Sobol' reeks verbeter die konvergensie van die beramer die effektiefste. Verder is die beste metode tussen Baanafhanklike Afgeleide Beramers en Eindige Differensie Benaderings om die sensitiwiteit vir die opsie te bepaal, ook gevind.

Deur dus die verfynde prysingsmetode met opsiepryse as Kontrole Variate, saam met Owen en Faure-Tezuke tipe verewekansigde Sobol' reekse as 'n Quasi-Monte Carlo metode te gebruik, word meer effektiewe metodes om die opsie te prys, gevind (in vergelyking met simplistiese Monte Carlo metodes). Verder is meer effektiewe sensitiwiteitsberamers as voorheen gevind wat gebruik kan word om risiko's te help bestuur.

ACKNOWLEDGEMENTS

The author would like to thank the following people for their contribution towards this project:

- Prof. Willie Conradie for the time, guidance and support he provided me during this project and during the rest of my studies - it is truly invaluable. Thank you.
- Mieke Fourie for all her contributions towards the structuring and proofreading.
- My internal promoter, Prof. Sarel Steel, for his inputs and guidance. As well as the rest of the staff, including Prof. Tertius de Wet, at the Department Statistics and Actuarial Science, who made the past two years absolutely unforgettable.
- Francois, Morné, Brigott and Retha for all their interest, encouragement and support.
- Wagner and Terence at Liberty for providing me the topic of this research and giving me the opportunity to present the findings to them.
- To all my friends and family who also made this journey a pleasant one.

CONTENTS

LIST OF ABBREVIATIONS

- AV - Antithetic Variable
- BS - Black-Scholes
- c.d.f. - cumulative distribution function
- CI - Confidence Interval
- CLT - Central Limit Theorem
- CV - Control Variate
- ERBCO - European Rebalanced Basket Call Option
- ERBO - European Rebalanced Basket Option
- ERBPO - European Rebalanced Basket Put Option
- FDA - Finite-Difference Approximation
- FS - Faure Sequence
- HS - Halton Sequence
- LDS - Low Discrepancy Sequences
- LHS - Latin Hypercube Sampling
- LLN - Law of Large Numbers
- i.i.d. - independently identically distributed
- IPIT - Inverse Probability Integral Transform
- MC - Monte Carlo
- NA - Not Applicable
- PA - Programmable Algorithm

- PDE - Pathwise Derivative Estimate
- p.d.f. - probability density function
- PIT - Probability Integral Transform
- QMC - Quasi-Monte Carlo
- RLDS - Randomised Low Discrepancy Sequences
- RMSE - Root Mean Squared Error
- RPD - Random Permutation of Digits
- RQMC - Randomised Quasi-Monte Carlo
- RRMSE - Relative Root Mean Squared Error
- RS - Random Shift
- SE - Standard Error
- SS - Sobol' Sequence
- VdC - Van der Corput
- VRT - Variance Reduction Technique

LIST OF ALGORITHMS

LIST OF DEFINITIONS

LIST OF FIGURES

LIST OF TABLES

LIST OF THEOREMS

CHAPTER 1

INTRODUCTION

Life insurance and pension funds offer a wide range of products that are invested in a mix of assets. The portfolios (Π) underlying these products are rebalanced every τ years back to predetermined fixed proportions. This is done by selling the better performing assets and buying the worse performing assets (these portfolios also exist widely in the unit trust market). Life insurance or pension fund contracts can offer the client a minimum payout guarantee on the contract by charging them an extra premium α. This guarantee forms an extra liability to the firm and needs to be valued on a daily basis and managed in terms of risks as well. It also helps with the management of the firm's reserve funds.

Therefore, given that the client will receive a payout of Π_T at the end of the life of the contract (the value of the portfolio at time T), this could be guaranteed to be a minimum of K. Therefore at time T the payout of this contract, which would have been Π_T, becomes $\max\{\Pi_T, K\}$. That is, $\Pi_T + \max\{K - \Pi_T, 0\}$ with the second part of this expression exactly the payoff of a put option. Therefore, this problem can be changed to that of the valuation of a European put option with underlying Π.

It follows that the insurance company is not only exposed to the portfolio Π, but also to an option on this portfolio. In practice these options are valued using simple Monte Carlo (MC) pricing methods. These liabilities are managed by studying the Greeks, also referred to as the option's sensitivities. The option's sensitivities give the risk manager an indication of how the liability reacts with regards to fluctuations of different parameters. Sensitivities are usually obtained by recalculating the value of a liability after a number of changes for some parameter. Such calculations take a significant amount of time, as each recalculation requires a different set of MC stochastic simulations.

In this thesis the value and sensitivities of this put option are estimated using different MC methods. These methods are compared to find the most efficient method. Only a put option with an underlying portfolio that consists of two assets is considered, but can easily be extended to more assets. Due to the large and increasing computational power of corporations' clusters of servers, simulation becomes a feasible numerical method for estimating prices and sensitivities of options where no

closed-form solution or formulae exist. Therefore the focus of this research will only be on MC numerical methods.

Although general methods to apply MC simulations to path-dependent and multi-asset options exist, currently no literature on this specific type of option exists. As such, this new option will from here on be known as the European Rebalanced Basket Call/Put Option (ERBCO/ERBPO), or in general the European Rebalanced Basket Option (ERBO).

Brigo et al. (2001), Curran (1994), Krekel et al. (2003), Deelstra et al. (2008), Milevsky and Posner (1998a, 1998b, 1999), Alexander and Venkatramanan (2009), Dhaene et al. (2002) and Ju (2002) all attempt to find closed-form solutions for either path-dependent or multi-asset options. The ERBO, however, is both path-dependent and has an underlying that depends on more than one asset (multi-asset). As such, only some of the information contained in these articles could be used for the purpose of this research.

In Prigent et al. (2004) the pricing of options on portfolios that are rebalanced after a fixed change in price occurred is considered. Krykova (2003) studied the valuation of path-dependent securities with low discrepancy methods. Boyle et al. (1997) gave the first in-depth insight for MC simulation techniques for option pricing. Kwok et al. (2001) proposed algorithms for the pricing of multi-asset path-dependent options and Wang (2001) gave new insights into variance reduction techniques (VRTs) for multivariate option pricing. These papers will be used selectively in this research.

A literature review on the definition and valuation framework for options is included as Appendix A. It provides the formal definition of options where stocks form the underlying of an option. Wiener Processes and Itô's lemma are used to derive the process followed by stock prices, which is used in both the MC pricing of options and the foundation of the Black-Scholes (BS) option pricing formulae for vanilla options. Chapter 2 provides a literature review on MC numerical methods. These include the general MC pricing method, VRTs, Quasi-Monte Carlo (QMC) methods and Sensitivity estimation. This literature will be applied in the chapters that follow.

The focus of Chapter 3 is on the valuation of the price of an ERBPO using general MC methods. A simplistic method is used initially, which is then improved substantially with the help of a new formula to simulate the value of the underlying portfolio. Chapters 4 and 5 improve the results obtained in Chapter 3 by incorporating VRTs and using QMC methods respectively. The different methods from each chapter are then compared to determine the most efficient method to price the ERBPO. Estimation of the sensitivities of the ERBPO will be discussed in Chapter 6, and the results will be compared to find the most efficient method.

1.1 RESEARCH ORIENTATION

Many numerical procedures exist to determine the price and sensitivities of options, including path-dependent or multi-asset options, but little research has been done on the pricing of combination

path-dependent multi-asset options. This lack is further aggravated by the fact that 'multi-asset' could mean anything from 2 to ∞ number of assets, and that 'path-dependent' is also very vague in terms of different types of path-dependent options.

1.1.1 Research Question

As such, the research question can be formulated as follows: what efficient approximations can be used to estimate the price and sensitivities of an option on an underlying that consists of two asset classes that are rebalanced on a regular basis using MC numerical methods?

1.1.2 Detailed description

Assume two assets with prices $S_{1,t}$ and $S_{2,t}$ at time t with volatility surfaces, risk-free rates, underlying processes and correlation between these two assets known. These assets make up the portfolio Π (with price Π_t at time t) with fixed proportions v_1 and $v_2 = 1 - v_1$. At the end of each period the portfolio is rebalanced. That is, the better performing asset is sold and the money is used to buy the poorer performing asset until the proportions are back to their fixed values v_1 and v_2. The value at any time for this portfolio will be denoted by Π_t. The parameters that will be used throughout this discussion are the following:

- T = time till expiration
- μ_i = expected growth of asset i per year
- r = the risk-free rate
- σ_i = volatility of asset i per year
- ρ = correlation between asset 1 and 2
- j = index for the j^{th} rebalance of the portfolio Π
- τ = rebalancing period (in years)
- $t = \tau j$ (any time can be expressed as the j^{th} rebalancing multiplied by the rebalancing period with $t \leq T$)
- t^* = the time immediately prior to the j^{th} rebalancing of the portfolio Π - that is $t^* = j\tau - \delta t$, with $\delta t \to 0$.
- $w_{i,j\tau}$ = the number of units of asset i that will be in the portfolio at rebalancing j, or any time up to the next rebalance

The processes followed by $S_{i,t}$ can be expressed as follows:

$$\frac{\mathrm{d}S_i}{S_i} = r\,\mathrm{d}t + \sigma_i \mathrm{d}z_i, \quad i = 1, 2;$$

with $E[\mathrm{d}z_1 \mathrm{d}z_2] = \rho$. Furthermore, at any time t the value of the portfolio can be expressed as

$$\Pi_t = w_{1,j\tau} S_{1,t} + w_{2,j\tau} S_{2,t},$$

with j being the time index of the rebalance prior to time t. In the context of this problem only the times of rebalancing, as well as the time immediately prior to rebalancing, i.e. times $t = j\tau$ and $t^* = j\tau - \delta t$ (with $\delta t \to 0$) are of interest. Therefore, assuming a person can hold fractions of an asset, the following holds with $\delta t \to 0$:

$$\begin{aligned} \Pi_{j\tau-\delta t} &= w_{1,(j-1)\tau} S_{1,j\tau-\delta t} + w_{2,(j-1)\tau} S_{2,j\tau-\delta t} \\ &= w_{1,j\tau} S_{1,j\tau} + w_{2,j\tau} S_{2,j\tau} \\ &= \Pi_{j\tau}. \end{aligned}$$

That is, the value of the portfolio just before the rebalancing takes place is exactly the same as the value of the portfolio afterwards, the only difference being the number of assets, w_1 and w_2, that are in the portfolio. These can be calculated with

$$w_{i,j\tau} = \frac{v_i \Pi_{j\tau-\delta t}}{S_{i,j\tau-\delta t}} = \frac{v_i \Pi_{j\tau}}{S_{i,j\tau}}, \quad j = 0, 1, ..., \left\lfloor \frac{T}{\tau} \right\rfloor; \; i = 1, 2. \tag{1.1}$$

For example, the initial amount of assets held can be computed with $w_{i,0} = \frac{v_i \Pi_0}{S_{i,0}}$. This is easily calculated as all inputs are known at $j = 0$.

The ERBO is an option on this portfolio with strike price K and time till maturity T, i.e. $\Psi_T \equiv \Pi_T$ with Ψ_T the value of the underlying at time T (as defined in Appendix A).[1] Thus, the discounted pay-off of the ERBPO is,

$$Y = \max\{K - \Pi_T, 0\} e^{-rT}, \tag{1.2}$$

[1] Only the put option will be considered in this research, but formulae and calculations can easily be altered to calculate the price and sensitivities of an ERBCO. These adaptations are included in Appendix D.

so that the price, or premium, is simply $E[Y] = \alpha$. This forms the core of all computations in the following chapters. Note also that $Y = H(\cdot)$ is a function of the different input parameters.

1.2 SUMMARY

Due to the nature of exotic options, little research has been done on specific types. As such, this research aims to contribute towards the efficient pricing of a specific type of exotic option and estimating its sensitivities, namely the ERBCO/ERBPO. The process followed by stocks forming the underlying of stock options, are key to the valuation of such options. The process followed by a stock was thus derived and can be used to price options. Important assumptions must be made for these valuation frameworks to hold and these are discussed in Appendix A.

CHAPTER 2

LITERATURE REVIEW

As stated in the previous chapter there exist several techniques to determine the premium that needs to be charged for the option by the individual holding the short position. These techniques will be the focus of this chapter. Firstly, general MC simulation as a numerical approach to pricing options for calculating the premium will be discussed. General MC can be improved by using VRTs, which form the topic of the second section. The third section introduces the concept of using low-discrepancy sequences (LDSs) instead of pseudo-random numbers in the simulation; this is called QMC simulation. In the last section, literature on estimating sensitivities of options using MC methods, is discussed. The literature review provides the theory needed to evaluate the calculation aspects of the ERBO. This chapter will be concluded by outlining the objectives of the rest of this research in terms of the literature review.

2.1 GENERAL MONTE CARLO SIMULATION

MC simulation uses the risk-neutral valuation result (as discussed in Appendix A.2) to value options. By sampling possible paths for the underlying to obtain the expected payoff in a risk-neutral world and then discounting this payoff at the risk-free rate, the premium that needs to be charged for an option can be estimated. In this section a general discussion of MC simulation will be provided, which will be supplemented by the simulation procedure for pricing a plain vanilla option. It is followed by a discussion on how this can be applied to exotic options and methods for evaluating the efficiency of different estimates. The next two subsections on the MC simulation framework and this framework in terms of plain vanilla options summarise the theory presented in Hull (2009:418-424), Alexander (2008a:217-219), Wilmott (2007:581-588) and Glasserman (2004:39-95).

2.1.1 Monte Carlo simulation framework

Let X be a given random variable with $E[X] = \lambda$, where the true value is unknown, and $Var(X) = \sigma^2$. In MC simulation, given the distribution of X, n independent observations of X i.e. $\{X_i : i = 1, ..., n\}$ are generated. The parameter λ is estimated by $\hat{\lambda}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$ - the sample mean of $\{X_1, ..., X_n\}$. This implies that $E[\hat{\lambda}(n)] = E[X] = \lambda$ making $\hat{\lambda}(n)$ an unbiased estimator for $E[X]$ with $Var(\hat{\lambda}(n)) = \frac{\sigma^2}{n}$. As the number of simulations n increases, $\hat{\lambda}(n)$ becomes a better estimate for λ - a consequence of the Law of Large Numbers (LLN):

Theorem 2.1 (The Law of Large Numbers) *Let $X_1, ..., X_n$ be independently identically distributed (i.i.d.) random variables with mean λ and variance σ^2. Then for any given $\delta > 0$, $P(|\hat{\lambda}(n) - \lambda| > \delta) \to 0$ as $n \to \infty$ (Rice, 2007:175).*

The general MC estimation method will be applied to plain vanilla type options in the next section.

2.1.2 Monte Carlo simulation of plain vanilla type options

MC simulation can be used to determine the premium for an option where the risk-neutral valuation assumption is used in pricing techniques, along with the results obtained for the process of $\ln S$ (see Appendix A.3). The process which must be followed to determine the price of a plain vanilla option with S_0, K, r, T and σ known, can be summarised as follows:

Algorithm 2.1 (Steps for MC simulation of a plain vanilla option)

1. *Sample random paths for S in a risk-neutral world.*
2. *Calculate the payoff from this derivative at time T.*
3. *Discount the simulated payoff with the risk-free rate back to time 0, this is a simulated value for the price of an option.*
4. *Repeat steps 1 to 3, n times to get n values for possible values of the option in the risk-neutral world.*
5. *Calculate the average of the n values in step 4: this is known as the estimate for the price of the option.*

Sampling of the stock price is the most important part of MC simulation. The necessary results have been derived (see Appendix A) and will be applied in this section. With the help of Itô's lemma the process followed by $\ln S$ was derived. The process for $\ln S$ had to be derived as this can be applied to obtain an exact distribution for $\ln S_T$ rather than an approximation. Assuming risk-neutral valuation, the expected growth rate for any stock can immediately be replaced with $\mu_i = r$ - the risk-free rate. The formula that is used to estimate S_T will now be derived.

From Itô's lemma it is known that

$$\mathrm{d}\ln S = \left(r - \frac{\sigma^2}{2}\right)\mathrm{d}t + \sigma\,\mathrm{d}z,$$

and discretising this in terms of Δt, the following is obtained:

$$\begin{aligned}
\Delta \ln S &= \left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\Delta z \\
\Delta \ln S &= \left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon\sqrt{\Delta t} \\
\ln S_{t+\Delta t} - \ln S_t &= \left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon\sqrt{\Delta t} \\
\ln S_{t+\Delta t} &= \ln S_t + \left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon\sqrt{\Delta t} \\
S_{t+\Delta t} &= S_t \exp\left(\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon\sqrt{\Delta t}\right) \qquad (2.1) \\
\Rightarrow S_T &= S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\epsilon\sqrt{T}\right) \qquad (2.2)
\end{aligned}$$

with $\epsilon \sim N(0,1)$.

An important point of note is that ϵ is generated with the help of the inverse probability integral transform (IPIT):

Definition 2.1 (Inverse Probability Integral Transform) *The probability integral transform (PIT) states that if X is a continuous random variable, with cumulative distribution function F_X and if $Y = F_X(X)$, then Y has a uniform distribution on $[0,1]$, i.e. $Y \sim Unif(0,1)$. The IPIT is just the inverse of this. Specifically, if $Y \sim Unif(0,1)$ and if X is defined as $X = F_X^{-1}(Y)$, then X has a cumulative distribution function F_X (Rice, 2007:352-358).*

The IPIT therefore implies that to simulate any $\epsilon \sim N(0,1)$ one only needs to simulate a random, uniformly distributed variable $U \sim Unif(0,1)$ and use this with the inverse of the cumulative distribution function for a standard normal distribution to simulate $\epsilon = F_\epsilon^{-1}(U)$ where $\epsilon \sim N(0,1)$ and $F_\epsilon^{-1}(\cdot) = \Phi^{-1}(\cdot)$ the inverse of the c.d.f. for a standard normal distribution.

Equation 2.2 can be interpreted as follows: given a random $N(0,1)$ variable ϵ, then a possible value for S_T can be calculated from Equation 2.2. This is the first step towards pricing a plain vanilla option - simulating paths for the underlying stock. The next step is to determine the payoff at time T.

Consider the case where the simulated price is S_T for the stock underlying the option. At time T, depending on whether it is a call or put, the payoff of this option will be either $\max\{S_T - K, 0\}$ or

$\max\{K - S_T, 0\}$. These values need to be discounted back to time 0 with the risk-free rate. Denote the discounted value by X. Step 4 states that values have to be simulated n times (the total number of simulations), generating $\{x_1, ..., x_n\}$. These x_i's are the n simulated values for the option price at time 0. The average of these, $\frac{1}{n}\sum_{i=1}^{n} x_i$, is the estimate for $E[X] = \lambda$, the premium of the option.[1]

2.1.3 Simulating path-dependent and multi-asset options

The concept of path-dependent and multi-asset options is introduced in Section A.1 in Appendix A. Hull (2009:591-607), Alexander (2008a:220-222), Wilmott (2007:594-596) and Glasserman (2004:96-107) all discuss this in terms of simulation and this section was adapted from their work. These two types of exotic options will now be discussed separately in terms of the MC simulation for plain vanilla options. Path-dependent options are similar to plain vanilla options. The only difference is that simulation requires the path followed by the underlying S. This approach would be best explained with the help of an example: say the payoff of an option depends on the average of the year-end closing prices of a stock over the past three years, Ψ_T. To calculate Ψ_T the simulation would require only $\{S_1, S_2, S_{T=3}\}$. It is thus only necessary to simulate those values, and therefore only for those jumps.[2] Therefore, when an option is path-dependent, it is necessary to simulate the underlying for those times upon which the final value of Ψ_T is dependent.

Looking back at the example in the previous paragraph one would first sample S_1 with the help of $S_1 = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)1 + \sigma\epsilon\sqrt{1}\right)$. This will be used to sample the next jump of this particular sample path: $S_2 = S_1 \exp\left(\left(r - \frac{\sigma^2}{2}\right)1 + \sigma\epsilon\sqrt{1}\right)$, etc. (each with newly generated ϵ's). Therefore, when using Equation 2.1, where Δt is the length for which the move is important, the path followed by the option can easily be obtained. The value Ψ_T in this example can then be calculated as $\Psi_T = \frac{S_1+S_2+S_3}{3}$ and the rest of the procedure follows as usual (see Steps 2 to 5).

Before considering simulation of a two-asset option, the process followed by the underlying stocks needs to be discussed. For multi-asset options the correlated underlying stocks are assumed to follow the following process (considering that in risk-neutral valuation all assets are assumed to have the same return, r):

$$\frac{\mathrm{d}S_i}{S_i} = r\,\mathrm{d}t + \sigma_i \mathrm{d}z_i \tag{2.3}$$

with

$$\hat{E}[\mathrm{d}z_i \mathrm{d}z_j] = \rho_{ij}\,\mathrm{d}t, \tag{2.4}$$

[1] Instead of discounting all the simulated payoffs individually and then taking the average, it is less time consuming to first calculate the average of the simulated payoffs and then discounting that value (the same results are obtained).

[2] For plain vanilla options the underlying share only jumps from S_0 to S_T.

with $\hat{E}$ the expected value in the risk-neutral world, and ρ_{ij} the correlation between stocks i and j. In this literature review the multi-asset option will be limited to a two-asset option, where ρ indicates the correlation between the two assets. In sampling the paths of these two assets the following can be used:

$$S_{1,T} = S_{1,0} \exp\left(\left(r - \frac{\sigma_1^2}{2}\right) T + \sigma_1 \epsilon_1 \sqrt{T}\right) \tag{2.5}$$

$$S_{2,T} = S_{2,0} \exp\left(\left(r - \frac{\sigma_2^2}{2}\right) T + \sigma_2 \epsilon_2 \sqrt{T}\right) \tag{2.6}$$

with

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix} \sim MVN_2\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}; \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\right).$$

To help overcome this obstacle of having to generate two correlated variables ϵ_1 and ϵ_2, one can make use of the following theorem:

Theorem 2.2 (Decomposition of two correlated normally distributed random variables) *Correlated random variables ϵ_1 and ϵ_2 can be decomposed into two uncorrelated random variables Z_1 and Z_2 through the linear transformation:*

$$\begin{aligned} Z_1 &= \epsilon_1 \\ Z_2 &= \frac{\epsilon_2 - \rho\epsilon_1}{\sqrt{1-\rho^2}} \end{aligned}$$

with Z_i independently distributed as $N(0,1)$, $i = 1, 2$.

The above result can be obtained with the use of Cholesky factorisation. It will be explained in terms of generating d correlated normally distributed variables $\epsilon_1, \epsilon_2, ..., \epsilon_d$. A sequence of d uncorrelated normally distributed variables $Z_1, Z_2, ..., Z_d$ can be generated and transformed with $\underline{\epsilon} = M\underline{Z}$, where $\underline{\epsilon}^T = (\epsilon_1, ..., \epsilon_d)$ and $\underline{Z}^T = (Z_1, ..., Z_d)$ are column vectors. The matrix $M : d \times d$ must satisfy $MM^T = \Sigma$, with $\Sigma : d \times d$ the correlation matrix.

This can be confirmed by taking the expectation of $\underline{\epsilon}\underline{\epsilon}^T = M\underline{Z}\underline{Z}^T M^T$:

$$E[\underline{\epsilon}\underline{\epsilon}^T] = ME[\underline{Z}\underline{Z}^T]M^T = MM^T = \Sigma.$$

It is possible to solve sequentially for the entries of M individually, where $M_{ij} \equiv (i,j)^{\text{th}}$ element of M and $\Sigma_{ij} \equiv (i,j)^{\text{th}}$ element of Σ, by using the following:

From the basic identity

$$\Sigma_{ij} = \sum_{k=1}^{j} M_{ik} M_{jk}, \quad j \leq i,$$

one obtains

$$M_{ij} = \left(\Sigma_{ij} - \sum_{k=1}^{j} M_{ik} M_{jk} \right) / M_{jj}, \quad j < i,$$

and

$$M_{ii} = \sqrt{\Sigma_{ii} - \sum_{k=1}^{i-1} M_{ik}^2}.$$

A simple algorithm based on the work of Golub and Van Load (as cited by Glasserman, 2004:73) that can be used to calculate the matrix M with the help of Cholesky factorisation, follows[3]:

Algorithm 2.2 (Cholesky factorisation)

Input: Σ - *Symmetric positive definite* $d \times d$ *matrix*
Output: M - *Lower triangular* M *with* $MM^T = \Sigma$
Algorithm:

$M \leftarrow 0$ *(*$d \times d$ *zero matrix)*
$\underline{v} \leftarrow \underline{0}$ *(vector of length* d*)*
for $j = 1, ..., d$
  for $i = 1, ..., d$
    $\underline{v}[i] \leftarrow \Sigma[i,j]$
    for $k = 1, ..., j-1$
      $\underline{v}_i \leftarrow \underline{v}_i - M[j,k] \times M[i,k]$
    end for
    $M[i,j] \leftarrow \underline{v}[i] / \sqrt{\underline{v}[j]}$
  end for
end for

■

[3]Or, use standard software for Cholesky decomposition, e.g. chol() in R (R Core Development Core Team, 2010).

The previous result thus implies that whenever one needs to generate ϵ_1 and ϵ_2, it can easily be done by only generating two independent, normally distributed, random variables $Z_i \sim N(0,1)$ and applying the following transformation:

$$\epsilon_1 = Z_1 \tag{2.7}$$
$$\epsilon_2 = Z_2\sqrt{1-\rho^2} + \rho Z_1. \tag{2.8}$$

Therefore, considering Equations 2.5 and 2.6, sample paths for two correlated stocks can easily be simulated, Ψ_T which depends on both S_1 and S_2 can be calculated, and the MC simulation can be executed as usual.

2.1.4 Valuating multi-asset and path-dependent options as integrals

MC simulation can be viewed as a problem of integral approximation. The expected value of a function can be calculated by integrating over the whole sample space. The simulation for multi-asset and/or path-dependent options can be regarded as a multi-dimensional integral. Krykova (2003:7-8) provides the following definition for evaluating a multidimensional integral:

Definition 2.2 (MC approximation for a multi-dimensional integral) *Given a random vector $\underline{X}$ having p.d.f. $f(\underline{x})$, then expectations in the form of $E[H(\underline{X})]$ can be approximated with MC simulation. The expression for the MC approximation for the multi-dimensional integral is given by*

$$\lambda = \int_S H(\underline{X}) f(\underline{X}) \mathrm{d}\underline{X} \approx \frac{1}{n} \sum_{\substack{i=1 \\ \underline{x} \in S}}^{n} H(\underline{x}_i)$$

where $\{\underline{x}_i : i = 1, ..., n\}$ are n independent random observations from the distribution of $\underline{X}$ that are obtained from n independent random observations $\underline{U}_i$ from a uniform distribution on $[0,1]^d$, where d is the dimension of the unit hypercube such that $S \in \mathbb{R}^d$.

The $\underline{U}_i$ mentioned in Definition 2.2 are regarded as pseudo-randomly generated, uniformly distributed, random variables in normal MC simulation. This will, however, change when looking at QMC simulation.

2.1.5 Efficiency of simulation estimators

A simple way of determining the best method for estimation is comparing the SE of the estimates. The smaller the SE, the more narrow the Confidence Interval (CI), and therefore the better the estimate. Unfortunately, this is not the only factor that needs to be considered; bias and computing time are also relevant. The two most important factors in terms of this research is error and

computing time, as all the estimators that will be discussed are unbiased (except for some sensitivity estimators). Glasserman (2004:9-18) provides an in depth explanation of the different ways in which estimation methods can be compared. This section is adapted from his work.

Before measures of efficiency can be discussed any further, two important results must be stated first. These results are used to assess the accuracy of $\hat{\lambda}(n)$. They are the LLN (see Theorem 2.1) and the Central Limit Theorem (CLT). The CLT implies that as n tends to infinity, the distribution of the random variable $\hat{\lambda}(n)$ behaves approximately like a normally distributed random variable.

Theorem 2.3 (Central Limit Theorem) *Let $X_1, ..., X_n$ be i.i.d. random variables with mean λ and variance σ^2. Then, as n tends to infinity*

$$P\left(\frac{\hat{\lambda}(n) - \lambda}{\sigma/\sqrt{n}} \leq z\right) \to \Phi(z),$$

where $\Phi(z)$ denotes the c.d.f. of a standard normal distribution evaluated at the point z (Rice, 2007:184).

Theorem 2.3 together with the sample variance $\hat{\sigma}^2 = \frac{1}{n-1}\sum(X_i - \hat{\lambda}(n))^2$ (an unbiased estimator for σ^2) is used to construct a CI for λ.

Definition 2.3 (Confidence Intervals) *If $\hat{\lambda}(n) = \bar{x}$, $\hat{\sigma} = s$, and $z_{\beta/2}$ the $\beta/2$ upper quantile of the standard normal distribution, then the interval $\left(\bar{x} - z_{\beta/2}\frac{s}{\sqrt{n}}; \bar{x} + z_{\beta/2}\frac{s}{\sqrt{n}}\right)$ is an approximate $100(1-\beta)\%$ CI for $\hat{\lambda}(n)$ (Glasserman, 2004:542).*

Ideally one would want a small β together with a small width for the CI. The CI is directly related to the size of the standard error (SE) $= \frac{\sigma}{\sqrt{n}}$ of the estimate, therefore if one can reduce the size of the SE then the interval will be smaller as well.

Now, in terms of options, let α be an unknown aspect of an option that needs to be estimated by $\hat{\alpha}(n)$, the corresponding unbiased estimator of α, that is $E[\hat{\alpha}(n)] = \alpha$. As an example, consider estimation of the expected price as an aspect of an option. Therefore P, the price of the option, is a random variable with $E[P] = \alpha$ and $Var(P) = \sigma_P^2$.

Now obtain a sample of n observations of P, say $P_1, ..., P_n$ with $E[P_i] = \alpha$ and $Var(P_i) = \sigma_P^2$ for all $i = 1, ..., n$, then the estimator for the price becomes

$$\hat{\alpha}(n) = \bar{P}(n) = \frac{1}{n}\sum_{i=1}^{n} P_i, \tag{2.9}$$

which is an unbiased estimator for α, because $E[\bar{P}(n)] = \alpha$. Furthermore, $Var(\bar{P}(n)) = \frac{\sigma_P^2}{n}$. Given a sample, $\bar{P}(n)$ is observed as $\bar{p}(n) = \frac{1}{n}\sum_{i=1}^{n} p_i$, where p_i is an observed value of P_i, then $\bar{p}(n)$ is an unbiased estimate of α.

Applied to options, the CLT implies that

$$P\left(\frac{\bar{P}(n) - E[P]}{\sigma_P/\sqrt{n}} < z\right) = P\left(\frac{\hat{\alpha}(n) - \alpha}{\sigma_\alpha/\sqrt{n}} < z\right) = \Phi(z), \tag{2.10}$$

as $n \to \infty$, with α being some aspect of the option. Alternatively,

$$\frac{\hat{\alpha}(n) - \alpha}{\sigma_\alpha/\sqrt{n}} \dot{\sim} N(0,1) \tag{2.11}$$

for large n. That is, the CLT gives information about the distribution of the error of the simulation estimate

$$\hat{\alpha}(n) - \alpha \dot{\sim} N(0, \sigma_\alpha^2/n). \tag{2.12}$$

Therefore the error on the left has approximately the distribution on the right. Thus, *ceteris paribus*, when comparing two estimators of the same quantity, the one with a smaller SE is preferred. When comparing two unbiased estimators, not only the SE plays a role, computational time is really important too. Glasserman (2004:10-12) shows that if one has a computational budget time q and a single replication P_i taking a fixed amount of computing time r, then the number of replications one can complete given the available budget is $\lfloor q/r \rfloor$, and the resulting estimator is $\hat{\alpha}(\lfloor q/r \rfloor)$ with

$$\hat{\alpha}(\lfloor q/r \rfloor) - \alpha \dot{\sim} N(0, \sigma^2/(q/r)) \tag{2.13}$$

when $q \to \infty$. Different MC methods will be compared according to the above.

This method however cannot be used for QMC techniques, as QMC focuses more on the convergence of the estimator rather than the reduction in variance (which will become clear later on). QMC techniques will be compared with the Root Mean Square Error (RMSE) over a fixed set of problem instances. That is, given m problems with true values $\alpha_1, ..., \alpha_m$ and n-point approximations $\hat{\alpha}_1(n), ..., \hat{\alpha}_m(n)$, the RMSE is

$$RMSE(n) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{\alpha}_i(n) - \alpha_i)^2}. \tag{2.14}$$

The smaller the RMSE, the quicker a method converges to the true value. The true values, however, are not always known, but this issue will be addressed in the chapter on QMC techniques. Another measure is the Relative RMSE (RRMSE),

$$RRMSE(n) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\hat{\alpha}_i(n) - \alpha_i}{\alpha_i}\right)^2}. \tag{2.15}$$

2.2 VARIANCE REDUCTION TECHNIQUES

In MC simulation, $\lambda = E[X]$ is estimated by generating a sample $\{X_i : i = 1, ..., n\}$ and then determining $\hat{\lambda}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$, furthermore the SE of the estimator $\hat{\lambda}(n)$ is $\frac{\sigma}{\sqrt{n}}$ with σ^2 being the variance of X. Note that there are two elements affecting the SE, namely $\sqrt{n}$ and σ. The first element can easily be interpreted: the more simulations that are done, the smaller the SE will become, and the more accurate the estimate will be. The other element is the square root of the variance of the simulated variable X. Therefore to make the SE smaller, the variance of X should be reduced. There exist several techniques to accomplish this called VRTs, which will be discussed in this section. Four VRTs will be discussed in this literature review, namely Antithetic Variables (AVs), Control Variates (CVs), Stratified Sampling and Latin Hypercube Sampling (LHS). Other techniques include Importance Sampling and Moment Matching Methods, but these fall beyond the scope of this literature review. In each of the following sections, a VRT will be discussed in terms of plain vanilla options and then modified to incorporate path-dependent options and multi-asset options.

2.2.1 Antithetic Variables

Chan and Wong (2006:89-90) describe AVs with a very simple example: suppose $\lambda = E[X]$ is to be estimated by generating two variables X and Y such that $E[X] = E[Y] = \lambda$ and $Var(X) = Var(Y) = \sigma^2$. Then $E\left[\frac{X+Y}{2}\right] = \lambda$ and

$$\begin{aligned} Var\left(\frac{1}{2}(X+Y)\right) &= \frac{1}{4}(Var(X) + Var(Y) + 2Cov(X,Y)) \\ &= \frac{\sigma^2}{2} + \frac{1}{2}Cov(X,Y) \end{aligned}$$

$$\leq \frac{\sigma^2}{2} \text{ if } Cov(X,Y) \leq 0.$$

This implies that if X and Y are negatively correlated then the variance of $\frac{1}{2}(X+Y)$ will be less than when they are independent, that is $\frac{\sigma^2}{2}$.

Definition 2.1 showed how ϵ's, which are distributed $N(0,1)$, could be generated with ease from uniformly distributed random variables on $(0,1)$. It can be shown that U_i and $V_i = 1 - U_i$ (both $\sim Unif(0,1)$) are negatively correlated. Now if H is any function, then $X = H(U_1, ..., U_d)$ will have the same distribution as $Y = H(V_1, ..., V_d)$, and a sample of the variables X and Y can be generated from only generating $\underline{\mathrm{U}}^T = (U_1, ..., U_d) \sim Unif(0,1)^d$. If H is a monotone function then X and Y will be negatively correlated.

Consider MC simulation for plain vanilla option prices. When reviewing the steps that needs to be taken to simulate the n sample values $\{X_1, ..., X_n\}$ of the discounted payoff, it is clear that these values all depend on a certain monotone function, say H, so that $X = H(\underline{\mathrm{U}})$ is the discounted payoff of an option. Now, consider another variable $Y = H(\underline{\mathrm{V}})$ where $\underline{\mathrm{V}} = 1 - \underline{\mathrm{U}}$. Then $E[X] = E[Y] = \lambda$ and $Var(X) = Var(Y) = \sigma^2$. Therefore, when simulating $\{X_1, ..., X_n\}$ with n $\underline{\mathrm{U}}_i$'s and $\{Y_1, ..., Y_n\}$ with n $\underline{\mathrm{V}}_i$'s, then each X_i and Y_i will be negatively correlated and the function H is monotone for $i = 1, ..., n$. Thus, $\lambda = E\left[\frac{X+Y}{2}\right]$ needs to be estimated. Thus the estimator for λ is

$$\hat{\lambda}_{AV}(n) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{H(\underline{\mathrm{U}}_i) + H(\underline{\mathrm{V}}_i)}{2}\right) \tag{2.16}$$

with

$$Cor(H(\underline{\mathrm{U}}), H(\underline{\mathrm{V}})) = \rho < 0.$$

Furthermore,

$$\begin{aligned} Var(\hat{\lambda}_{AV}(n)) &= \frac{1}{4n^2}\sum_{i=1}^{n}\left(Var(H(\underline{\mathrm{U}}_i)) + Var(H(\underline{\mathrm{V}}_i)) + 2Cov(H(\underline{\mathrm{U}}_i), H(\underline{\mathrm{V}}_i))\right) \\ &= \frac{1}{4n^2}(2n\sigma^2 + 2n\rho\sigma^2) \\ &= \frac{\sigma^2}{2n}(1+\rho). \end{aligned}$$

Finally, the algorithm for MC simulation for plain vanilla options needs to be adjusted to incorporate AVs. The simulation is divided into two parts.

Algorithm 2.3 (Steps for MC simulation with AV as a VRT)

1. *Complete normal MC simulation with random uniformly distributed variables (U_is) on* $(0, 1)$.
2. *For the second part take the U_is that were used in the previous step and calculate $V_i = 1 - U_i$. Complete normal simulation with these uniformly distributed random variables.*
3. *Two sets of identically distributed sample values for the discounted payoff for the simulated option now exist which should theoretically be negatively correlated.*
4. *Estimate $\lambda = E\left[\frac{X+Y}{2}\right]$ with Formula 2.16.* ■

Note that, the reduction in variance for AV should be compared to simple MC using $2n$ repetitions. Furthermore, it is important that $H(\cdot)$ should be a monotone function of each of its arguments.

The SE for this result obtained using AVs will be less than when it is not applied. For path-dependent options, simply make sure that if there are m jumps that $X = H(U_1, ..., U_m)$ and $Y = H(V_1, ..., V_m)$ with $V_i = 1 - U_i$ for every simulated observation. The same holds for multi-asset options, with $m =$ the number of assets.

2.2.2 Control Variates

CVs are discussed in Chan and Wong (2006:104-109), Huynh et al. (2008:79-80) and Glasserman (2004:185-196), and will be summarised here. When $\lambda = E[X]$ is estimated with MC simulation another variable called the control variate Y can be introduced. This variable has a known mean, $\mu_Y = E[Y]$. For any given constant c, the quantity $X_{CV} = X + c(Y - \mu_Y)$ can be used to construct an unbiased estimator for λ as $E[X_{CV}] = \lambda$. The estimator becomes

$$\hat{\lambda}_{CV}(n) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i + c\left((Y_i - \mu_Y)\right)\right). \tag{2.17}$$

Consider the variance of X_{CV}, $\sigma^2_{CV} = Var(X + c(Y - \mu_Y)) = Var(X) + c^2 Var(Y) + 2cCov(X, Y)$. The next step is to find a c that minimises σ^2_{CV}, called c^*. By taking the derivative of σ^2_{CV} with respect to c and equalling it to zero, such a c can easily be found:

$$\begin{aligned} \frac{\mathrm{d}\sigma^2_{CV}}{\mathrm{d}c} &= 0 \\ 2cVar(Y) + 2Cov(X, Y) &= 0 \\ \Rightarrow c^* &= -\frac{Cov(X, Y)}{Var(Y)}. \end{aligned} \tag{2.18}$$

Substituting Equation 2.18 back into the formula for σ^2_{CV} leads to:

$$\begin{aligned}\sigma^2_{CV(c^*)} &= Var(X) - \frac{Cov^2(X,Y)}{Var(Y)} \\ &= Var(X)(1 - Cor^2(X,Y))\end{aligned}$$

Therefore, as long as $Cor(X,Y) \neq 0$, the variance will be reduced. Note that $Var(Y)$ and $Cov(X,Y)$ are usually not available, and therefore must be estimated before the final simulation is done, so that the value of c^* can be determined:

$$\begin{aligned}\widehat{Cov}(X,Y) &= \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) \\ \hat{\sigma}^2_Y &= \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \\ \bar{X} &= \frac{1}{n}\sum_{i=1}^{n} X_i \\ \bar{Y} &= \frac{1}{n}\sum_{i=1}^{n} Y_i\end{aligned}$$

When considering the choice of the CV, it is important that firstly μ_Y should be known, and secondly that the CV needs to have high correlation with the simulated variate X.

For the simulation of plain vanilla options, the price of the underlying asset itself at time T is usually taken as the CV, with $E[S_T] = S_0 e^{rT}$ and $Var(S_T) = S_0^2 e^{2rT}\left(e^{\sigma^2 T} - 1\right)$. Therefore the only value that needs to be estimated before calculating c^* is $Cov(X, S_T)$. The procedure to incorporate CVs with the MC simulation for a plain vanilla option will be given below, with the CV being the stochastic variable S_T. Note that initially n_1 simulations will be performed to help calculate c^* which will be followed by n_2 simulations to estimate the price of the option.

Algorithm 2.4 (Steps for MC simulation with CVs)

1. *For $i = 1, ..., n_1$ simulate n_1 independent paths and obtain a sample of S_T values (Y); using these, calculate and obtain a sample of discounted payoffs from the option C (X).*
2. *Calculate c^* using Equation 2.18.*
3. *Repeat simulation as normal, simulate samples of S_T and calculate the discounted payoffs from it.*
4. *Calculate the average of n $C_{CV,i} = C_i + c^*(S_{T,i} - E[S_T])$, where $E[S_T] = S_0 e^{rT}$.* ■

For exotic options (path-dependent and/or multi-asset options) one can use the value of a plain vanilla option as a CV because closed-form solutions exist to calculate μ_Y - the BS formulae (see Appendix A.4).

2.2.3 Stratified Sampling

Stratified sampling originates from the fact that a function might be more or less variable at certain points in a simulation. To accommodate this fact, one can complete more simulations in the area where the function is more variable. Therefore the SE of those areas will be smaller (due to the increase of the amount of simulations) and the areas which are less variable will not be affected significantly by the reduced number of simulations.

This method can easily be applied to simulate a plain vanilla option by dividing $[0, 1]$ into B bins (or strata) and simulating N sample paths from each of them. Unfortunately, this becomes less practical when working with path-dependent options or multi-asset options.

For example, if this method was to be applied to a European single asset option which is path-dependent (say at 3 discrete times) then one would divide the first jump into B strata, each where N/B sample paths would be generated for the first jump. Each of those B strata need to be divided into further B strata for the next jump, and sample N/B^2 from each of them. The logic holds for the last jump. That renders B^3 final strata and N final samples.

Considering a two-asset option it becomes clear that for each strata simulated from for the first asset, one would need to simulate from all B strata for the other variable. When looking at path-dependent multi-asset options, this method becomes tedious and the increase in computational time might not justify the little reduction in variance achieved. This problem can be avoided by using LHS, discussed in the next section (Glasserman, 2004:209-210; Chan amd Wong, 2006:97-104).

2.2.4 Latin Hypercube Sampling

LHS is an extension of stratification, introduced in the previous section, for sampling in multi-dimensions. Glasserman (2004:209-210) explains that the difficulty arises even for the most simple case when sampling from the d-dimensional unit hypercube $[0, 1]^d$. Partitioning each coordinate into B strata produces B^d strata for the unit hypercube, thus requiring a sample size of at least B^d to ensure that each stratum is sampled.

The method for LHS was first introduced by McKay et al. (1979) and further analyzed in Stein (1987). The method is best described by Glasserman (2004:236-238) and the literature review on this method will be adapted from his work.

LHS treats all coordinates equally and avoids the exponential growth in sample size resulting from full stratification by stratifying only the one-dimensional marginals of a multi-dimensional joint distribution. Glasserman describes the method in the case of sampling from the uniform distribution

over the unit hypercube. Fix the dimension d and a sample size B. For each coordinate $i = 1, ..., d$, independently generate a stratified sample $V_i^{(1)}, ..., V_i^{(B)}$ from the unit interval using B equally probable strata; each $V_i^{(j)}$ is uniformly distributed over $[(j-1)/B, j/B]$. If the d stratified samples are arranged in columns,

$$\begin{array}{cccc} V_1^{(1)} & V_2^{(1)} & \cdots & V_d^{(1)} \\ V_1^{(2)} & V_2^{(2)} & \cdots & V_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ V_1^{(B)} & V_2^{(B)} & \cdots & V_d^{(B)} \end{array}$$

then each row gives the coordinates of a point in $[0,1]^d$. The first row identifies a point in $[0, 1/B]^d$, the second a point in $[1/B, 2/B]^d$, etc. This corresponds to B points falling in subcubes along the diagonal of the unit hypercube. Now, randomly permute the entries in each column of the array. More precisely, let $\pi_1, ..., \pi_d$ be random permutations of $\{1, ..., B\}$, where all $B!$ permutations are equally likely. Let $\pi_j(i)$ denote the value to which i is mapped by the j^{th} permutation. The rows of the array,

$$\begin{array}{cccc} V_1^{\pi_1(1)} & V_2^{\pi_2(1)} & \cdots & V_d^{\pi_d(1)} \\ V_1^{\pi_1(2)} & V_2^{\pi_2(2)} & \cdots & V_d^{\pi_d(2)} \\ \vdots & \vdots & \ddots & \vdots \\ V_1^{\pi_1(B)} & V_2^{\pi_2(B)} & \cdots & V_d^{\pi_d(B)}, \end{array}$$

continue to identify points in $[0,1]^d$, but they are no longer restricted to the diagonal. Indeed, each row is a point uniformly distributed over the unit hypercube. The B points determined by the B rows are not independent: if the B points are projected onto their i^{th} coordinates, a set of values $\{V_i^{\pi_i(1)}, ..., V_i^{\pi_i(B)}\}$ which is the same as the set $\{V_i^{(1)}, ..., V_i^{(B)}\}$ are obtained, and thus forms a stratified sample from the unit interval.

To generate an LHS of size B in dimension d, let $U_i^{(j)}$ be independent $Unif[0,1]$ random variables for $i = 1, ..., d$ and $j = 1, ..., B$. Let $\pi_1, ..., \pi_d$ be independent random permutations of $\{1, ..., B\}$ and set

$$V_i^{(j)} = \frac{\pi_i(j) - 1 + U_i^{(j)}}{B}, \quad i = 1, ..., d, \quad j = 1, ..., B.$$

For properties of LHS that shed light on its effectiveness, including the variance reduction and variance decomposition, see Glasserman (2004:240-241).[4] The package `LHS` (Carnell, 2009) for the statistical program `R` (R Development Core Team, 2010), has a built-in function for generating an LHS of size B for dimension d.

[4]Other literature includes Huynh et al. (2008:101-107) and Krykova (2003:23-26).

2.3 QUASI-MONTE CARLO TECHNIQUES

The previous two sections provided a literature review on general MC techniques and how VRTs can be used to make these more efficient. The basis of estimating a multi-dimensional integral is to generate a sample of the vector $\underline{\mathrm{U}}_i$, with $\underline{\mathrm{U}}_i$ generated from the d-dimensional unit hypercube $[0,1]^d$ (see Definition 2.2). In general MC, these are generated using a pseudo-random number generator.

The method of QMC simulation changes the randomness of simple MC by using deterministically chosen sequences of numbers instead of totally random sequences. This is the fundamental part of the QMC method and these deterministic sequences are called LDSs. Glasserman (2004:281) states that low-discrepancy methods have the potential to accelerate convergence from the $O(1/\sqrt{n})$ rate associated with MC (n being the number of simulations) to nearly $O(1/n)$ convergence. This section of the literature review will start by discussing the general principles of QMC techniques. Then generation of the four main types of LDSs will be discussed, followed by methods to randomise these sequences to sometimes obtain better results.

2.3.1 General Principles

As shown earlier, the price of an option can be estimated by using Definition 2.2. That is, QMC approximates

$$\lambda = E[H(\underline{\mathrm{U}})] = \int_{[0,1]^d} H(\underline{\mathrm{U}})\mathrm{d}\underline{\mathrm{U}}$$

by

$$\hat{\lambda}(n) = \frac{1}{n}\sum_{i=1}^{n} H(\underline{\mathrm{U}}_i)$$

so that for $n \to \infty$,

$$\hat{\lambda}(n) \to \lambda,$$

with deterministically chosen values of $\{\underline{\mathrm{U}}_i : i = 1, ..., n\}$ from the unit hypercube $[0,1]^d$. Note that the function H only needs to be evaluatable, it does not need to have an explicit form. Furthermore, in standard MC simulation it does not matter whether 0 and 1 are included in the simulation, but for certain QMC definitions and results it does matter. The norm is $[0,1)^d$ but when these are transformed to normal variables it needs to exclude both 0 and 1, i.e. $(0,1)^d$.

In normal MC, sequences from the d-dimensional hypercube $\underline{\mathrm{U}}_1 = (U_1, ..., U_d), \underline{\mathrm{U}}_2 = (U_{d+1}, ..., U_{2d})$ etc. are obtained by splitting up a series of $n \times d$ i.i.d. $U \sim Unif(0,1)$ distributed variables in vectors

of length d. In QMC the values for $\underline{U}_i$ depend explicitly on the dimension d. Therefore, without a proper upper bound for the dimension d, QMC is inapplicable.

2.3.2 Low-Discrepancy Sequences

Glasserman (2004:283-284) gives an intuitive explanation of discrepancy, summarised here. The most simplistic way to fill the unit hypercube uniformly is to choose the points $\underline{U}_i$ to lie on a grid. A grid, however, requires the number of points n in advance. If one wants to refine this grid by adding points, the number points that must be added to reach the next favourable configuration grows very quickly. LDSs overcome this by guaranteeing the uniformity over bounded-length extensions of an initial segment of the sequence.[5]

Krykova (2003:10-11) provides a list of advantages of QMC. Firstly, the integral is approximated using a well-chosen sequence of points. Secondly, the QMC approach often leads to better point estimates for similar computational efforts compared with standard MC. Quasi-random numbers result in faster convergence; fewer quasi-random samples are needed to achieve a similar level of accuracy as obtained by pseudo-random sequences; and finally, in several cases QMC permits improvement of the performance of MC simulations, offering shorter computational times and/or higher accuracy. The superior accuracy of QMC methods provide a way to improve the accuracy and reliability of MC simulation. Krykova (2003) also compares the valuation of path-dependent securities with low-discrepancy methods.

2.3.2.1 Van der Corput

Krykova (2003:12-13), Huynh et al. (2008:92-93) and Glasserman (2004:285-287) all provide definitions and algorithms for the Van der Corput (VdC) sequence, and a summary here will be adapted from their work.[6]

The VdC sequence is the basic (one-dimensional) sequence of many QMC methods. The method for determining the n^{th} element of the sequence will be given here. Before proceeding, the term 'base' needs to be defined first. A base is an integer, $b \geq 2$. Every positive integer k has a unique representation (called its base-b or b-ary expansion) as a linear combination of nonnegative powers of b with coefficients in $\{0, 1, ..., b-1\}$. The most notable type is for $b = 2$ which translates into binary expressions.

[5]For more literature on the mathematics behind LDSs, see Glasserman (2004:283). This however, is beyond the scope of this research.

[6]For proof of why the VdC sequence is an LDS, see Theorem 3.6 in Niederreiter (1992).

To find the n^{th} element the following operations are necessary:

Algorithm 2.5 (Van der Corput sequence)

1. *Write n in base-b; this allows one to find $a_j(n)$ such that:*

$$n = \sum_{j=0}^{m} a_j(n) b^j,$$

where m is the smallest integer such that $a_j(n) = 0$ for all $j > m$.

2. *Reverse the number n to the decimal point to find the value of the n^{th} element, which is denoted by b_n:*

$$U_n = \phi_b(n) = \sum_{j=0}^{m} \frac{a_j(n)}{b^{j+1}} \tag{2.19}$$

In reversing the number, one can make sure that the value lies in the interval (0,1). For example if $n = 11$ (base 10), or 102_3, then $3_n = 19/27$ which equals 0.201_3 (Huynh et al., 2008:92). ■

The VdC sequence is a special case of the Halton sequence (HS) with $d = 1$ (see the next subsection), therefore a VdC sequence can be generated with the `runif.halton()` function in the package `fOptions` (Wuertz, 2010) found in `R` (R Development Core Team, 2010) by setting $d = 1$.

2.3.2.2 Halton

As stated in the previous subsection: the HS is a multi-dimensional extension of the VdC sequence. Halton (1960), building on the work of Hammersley (1960), provides the simplest construction and first analysis of LDS in arbitrary dimension d. This method is easily derived from the VdC method. To build the HS one uses the points in the sequence of the VdC but change the base for each dimension, i.e. use base 2 for the first dimension, base 3 for the second dimension, base 5 for the third dimension, etc. The i^{th} dimension will be the VdC sequence obtained from the i^{th} prime number. Put more formally,

Definition 2.4 (Halton Sequences)

Let $b_1, ..., b_d$ be relatively prime integers greater than 1, and set

$$\underline{U}_k = (\phi_{b_1}(k), \phi_{b_2}(k), ..., \phi_{b_d}(k)), \quad k = 0, 1, 2, ..., n$$

with $\phi_b(k)$ the inverse function defined in Equation 2.19 (Glasserman, 2004:293). Then $\underline{U}_1, ..., \underline{U}_n$ forms the HS. ■

The requirement that the b_i be relatively prime is necessary for the sequence to fill the unit hypercube. Note that if $k = 0$, the first value for the HS will be $\underline{U}_0 = \underline{0}$. When the points are fed into a simulation algorithm, there is often good reason to avoid zero, for example $\Phi^{-1}(0) = -\infty$. Therefore in practice it is often not considered. Another point to note is that because the base of the VdC sequence gets larger as the dimension increases, it takes increasingly longer to fill the unit hypercube (for example, the 25^{th} and 26^{th} primes are 97 and 101 respectively) (Krykova, 2003:13).[7]

Glasserman (2004:296) also provides an algorithm for generating a HS.[8] As stated in the previous subsection, one can simply use the function `runif.halton()` in the `fOptions` package (Wuertz, 2010) in `R` (R Development Core Team, 2010) to generate a HS. The next sequence that will be discussed is the Faure sequence (FS) which overcomes the problem the HS experiences in high dimensions.

2.3.2.3 Faure

Faure (1982) developed a different extension of the VdC sequence to multiple dimensions. The FS is almost similar to the HS, but with two major differences: firstly, it uses only one base for all dimensions; and secondly, it uses a permutation of the vector composed of elements for each dimension. The base of an FS is the smallest prime number that is larger than or equal to the number of dimensions in the problem, or equal to 2 for a one-dimensional problem.

As occurred with high-dimensional HS, there is the problem of low speed at which the FS generates increasingly finer grid points to cover the unit hypercube. However, this problem is not too severe, as with the HS. For example, if the dimension of the problem is 50, the last HS (in dimension 50) uses the 50^{th} prime number (229), whereas the FS uses the first prime number after 50, that is base 53, which is much smaller than 229. By reordering the sequence within each dimension, FS prevents some problems of correlation for sequential high-dimensions that occurred with the HS. For a d-dimensional simulation, the VdC sequence in base-b is taken and the terms are permuted (where $b \geq d$). The algorithm is given by Glasserman (2004:298) as:

Algorithm 2.6 (Generating an FS)

Find the coefficients in the base-b expansion of k so that

$$k = \sum_{\ell=0}^{\infty} a_\ell(k) b^\ell.$$

The i^{th} coordinate, $i = 1, ..., d$, of the k^{th} point in the Faure sequence is given by

$$\sum_{j=1}^{\infty} \frac{y_j^{(i)}(k)}{b^j},$$

[7] For a comparison of how the HS performs at high dimensions, see Krykova (2003:14-16), Glasserman (2004:295-296) and Huynh et al. (2008:94).

[8] For proof on why the HS is an LDS see Glasserman (2004:294-295).

where

$$y_j^{(i)}(k) = \sum_{\ell=0}^{\infty} \binom{\ell}{j-1} (i-1)^{\ell-j+1} a_\ell(k) \mod b.$$

Each of the sums in the above three equations only have a finite number of nonzero terms. Suppose the base-b expansions of k in the first equation has exactly r terms, meaning that $a_{r-1}(k) \neq 0$ *and* $a_\ell = 0$ *for all* $\ell \geq r$*. Then the summands in the third equation vanish for* $\ell \geq r$*. If* $j \geq r+1$*, then the summands for* $\ell = 0, ..., r-1$ *also vanish, so* $y_j^{(i)}(k) = 0$ *if* $j \geq r+1$*, which implies that the second equation has at most r nonzero terms (Glasserman, 2004, 298).* ■

For a proof on why this is deemed an LDS and a general algorithm on how to generate an FS in simulation, see Glasserman (2004:300-302). The function, `runif.faure()` in the `DiceDesign` package (Franco et al., 2010) found in `R` (R Development Core Team, 2010), will be used to generate an FS.

2.3.2.4 Sobol'

Sobol' (1967) gave the first construction of what is known as a (t, d)-sequence. Sobol's construction can be contrasted with Faure's as follows: whereas Faure-points are $(0, d)$-sequences in a base at least as large as d, Sobol'-points are (t, d)-sequences in base 2 for all d, with values of t that depend on d. Faure-points therefore achieve the best value of the uniformity parameter t, but Sobol'-points have the advantage of a much smaller base. Working in base 2 also lends itself to computational advantages through bit-level operations.

The Sobol' sequence (SS) is generated using a set of so-called direction numbers

$$v_i = \frac{m_i}{2^i}, \quad i = 1, 2, ...$$

where $m_i \equiv$ odd positive integers $< 2^i$. The values of m_i are chosen to satisfy a recurrence relation using the coefficients of a primitive polynomial of order 2. A primitive polynomial is irreducible (i.e. it cannot be factored into polynomials of smaller degree) and does not divide by the polynomial $x^r + 1$ for $r < 2^p - 1$.

Corresponding to a primitive polynomial

$$P(x) = x^q + a_1 x^{q-1} + ... + a_{q-1} x^1 + 1$$

is the recursion

$$m_i = 2a_1 m_{i-1} \oplus 2^2 a_2 m_{i-2} \oplus ... \oplus 2^q a_q m_{i-q}$$

where $\oplus$ denotes binary addition.

For more information on the SS see Glasserman (2004:303-316) and Huynh et al. (2008:97-101) which includes general algorithms that can be applied in programming, and proofs on why this is an LDS. The function `runif.sobol()` in the `fOptions` package (Wuertz, 2010) in `R` (R Development Core Team, 2010) will be used to generate an SS.

2.3.3 Randomised Quasi-Monte Carlo

Glasserman (2004:320-321) provides reasons for the use of Randomised QMC (RQMC). Firstly, by randomising QMC points one opens the possibility of measuring error through a CI while preserving much of the accuracy of pure QMC. RQMC thus seeks to combine the best feature of ordinary MC with QMC. The second is that there are settings in which randomisation actually improves accuracy.

One RQMC technique will be discussed briefly in this section, namely Random Shift (RS). Others include: Random Permutation of Digits, Scrambled Nets and Linear Permutation of Digits. RS is the simplest of all RQMC where the point set $P_n = \{\underline{x}_1, ..., \underline{x}_n\}$ (each $\underline{x}_i \in [0,1)^d$) is shifted by adding a generated random vector $\underline{U}$ uniformly distributed over the d-dimensional unit hypercube modulo 1, i.e.

$$P_n(\underline{U}) = \{\underline{x}_i + \underline{U} \bmod 1;\ i = 1, ..., n\}.$$

The statistical package `R`'s function `runif.sobol` gives an option for scrambling of the Sobol' sequence. These include the Owen type as well as Faure-Tezuka type scrambling, and an option to apply both. The mathematics behind these scramblings is beyond the scope of this research, but the property remains the same - by randomising or scrambling a sequence, the possibility of measuring error through a CI becomes possible (while preserving much of the accuracy).[9]

2.4 SENSITIVITY ANALYSIS

The previous sections addressed aspects of estimating expectations with a view toward computing the prices of derivative securities. This section reviews methods for estimating sensitivities of expectations - in particular, the derivatives of derivative prices commonly referred to as 'Greeks'. Some literature on estimating sensitivities with MC methods exists, the most in depth literature being Glasserman (2004:377-420) and Huynh et al. (2008:207-219). This review on sensitivity analysis will be adapted from their work. The methods for estimating sensitivities that will be discussed, fall into two broad categories: methods that involve simulating at two or more values of the parameter of differentiation and methods that do not.

Glasserman (2004:377) explains that Finite-Difference Approximation (FDA) falls into the first category. It is an easy method to understand and implement, but because it produces a biased estimate

[9] See Owen (1998) and Tezuka and Faure (2003).

its use requires balancing bias and variance. The methods in the second category produce unbiased estimates. They accomplish this by using information about the simulated stochastic process to replace numerical differentiation with exact calculations. The Pathwise Derivative Estimate (PDE) method differentiates each simulated outcome with respect to the parameter of interest, while the likelihood ratio method differentiates a probability density rather than an outcome (but these will not be discussed). All methods will be discussed in general, and will be applied to the ERBPO in Chapter 6.

2.4.1 Finite-Difference Approximations

Consider the discounted payoff of an option $H(\theta)$, with the price of the option $\alpha(\theta)$ and θ any of the many model or market parameters that influence the price, then $\alpha(\theta) = E[H(\theta)]$. When θ is the initial price of an underlying asset, then $\alpha'(\theta) = \frac{\mathrm{d}\alpha(\theta)}{\mathrm{d}\theta}$ is the option's delta. The second derivative $\alpha''(\theta) = \frac{\mathrm{d}^2\alpha(\theta)}{\mathrm{d}\theta^2}$ is the option's gamma. When θ is a volatility parameter, $\alpha'(\theta)$ is the option's vega. When θ is the time till maturity, $\alpha'(\theta)$ is the option's theta; and finally, when θ is the risk-free rate, then $\alpha'(\theta)$ is the option's rho.

2.4.1.1 Bias and Variance

Glasserman (2004:378-379) gives the following approach to FDA: simulate independent replications $H_1(\theta), ..., H_n(\theta)$ of the model at parameter θ and n additional replications $H_1(\theta + h), ..., H_n(\theta + h)$ at $\theta + h$, for some $h > 0$. Then average each set of replications to get $\hat{\alpha}(\theta, n) = \bar{H}(\theta, n)$ and $\hat{\alpha}(\theta + h, n) = \bar{H}(\theta + h, n)$ and form the forward-difference estimator

$$\widehat{\alpha'}_F(\theta, h, n) = \frac{\hat{\alpha}(\theta + h, n) - \hat{\alpha}(\theta, n)}{h} \tag{2.20}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{H_i(\theta + h) - H_i(\theta)}{h}\right). \tag{2.21}$$

This estimator has expectation

$$E[\widehat{\alpha'}_F(\theta, h, n)] = h^{-1}[\alpha(\theta + h) - \alpha(\theta)]. \tag{2.22}$$

In terms of variance the relation between the outcomes $H_i(\theta)$ and $H_i(\theta + h)$ needs to be considered.

2.4.1.1.1 Bias

If α is twice differentiable at θ, then by taking the Taylor expansion,

$$\alpha(\theta + h) = \alpha(\theta) + \alpha'(\theta)h + \frac{1}{2}\alpha''(\theta)h^2 + o(h^2).$$

In this case, it follows from Equation 2.22 that the bias in the forward-difference estimator is

$$Bias\left(\widehat{\alpha'}_F(\theta, h, n)\right) = E[\widehat{\alpha'}_F(\theta, h, n) - \alpha'(\theta)] = \frac{1}{2}\alpha''(\theta)h + o(h). \tag{2.23}$$

By simulating at $\theta - h$ and $\theta + h$ one can form a central-difference estimator

$$\widehat{\alpha'}_C(\theta, h, n) = \frac{\hat{\alpha}(\theta + h, n) - \hat{\alpha}(\theta - h, n)}{2h} \tag{2.24}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{H_i(\theta + h) - H_i(\theta - h)}{2h}\right). \tag{2.25}$$

Similarly, if α is at least twice differentiable in the neighbourhood of θ, then by taking the Taylor expansion, it can be showed that $Bias(\widehat{\alpha'}_C(\theta, n, h)) = o(h)$, which is of smaller order than in Equation 2.23. This can be refined if α'' itself is differentiable at θ, then

$$Bias(\widehat{\alpha'}_C(\theta, n, h)) = \frac{1}{6}\alpha'''(\theta)h^2 + o(h^2). \tag{2.26}$$

2.4.1.1.2 Variance

The form of the bias of the forward- and central-difference estimators gives one good reason for taking small values of h to improve accuracy, but unfortunately the effect of h on bias must be weighed against its effect on variance. The variance of the forward-difference estimator is

$$Var(\widehat{\alpha'}_F(\theta, h, n)) = h^{-2}Var(\hat{\alpha}(\theta + h, n) - \hat{\alpha}(\theta, n)) \tag{2.27}$$

and a corresponding expression holds for the central-difference estimator in Equation 2.25. It is clear that when h is small the variance increases exponentially in h when h decreases. The above equation also shows that the dependence between values simulated at different values of θ affects the variance of a finite-difference estimator.

2.4.1.2 Optimal Mean Square Error

Glasserman (2004:381-386) gives a summary of the literature on finding the optimal h by minimising the mean square error (MSE). The problem remains that, by taking a smaller h it increases variance while decreasing bias (and vice versa). Minimising MSE requires balancing these two considerations. Another parameter to consider is the number of replications, n, which only decreases variance and has no effect on bias for larger n. This section attempts to find the optimal relation between n and h.[10]

Glasserman (2004:382) presents Table 2.1 as the results for optimal h. He states that the table should be understood as follows: if the leading terms of the variance and bias are as indicated in the second and third columns, then the conclusions in the last three columns hold. The results in the table indicate that, at least asymptotically, $\widehat{\alpha}_{C,ii}(\theta, h, n)$ dominates the other three estimators because it exhibits the fastest convergence. Only the final results will be given here.[11]

Table 2.1: Convergence rates of finite-difference estimators with optimal increment h_n. The estimators use either forward (F) or central (C) differences and either independent sampling (i) or common random numbers (ii). Source: Glasserman, 2004:382.

Estimator	Variance	Bias	Optimal h_n	Convergence	h_*
$\widehat{\alpha}_{F,i}(\theta,h,n)$	$\frac{\sigma^2_{F,i}}{nh^2}$	$\frac{1}{2}\alpha''(\theta)h$	$O(n^{-1/4})$	$O(n^{-1/4})$	$\left(\frac{4\sigma^2_{F,i}}{\alpha''(\theta)^2}\right)^{1/4}$
$\widehat{\alpha}_{C,i}(\theta,h,n)$	$\frac{\sigma^2_{C,i}}{nh^2}$	$\frac{1}{6}\alpha'''(\theta)h^2$	$O(n^{-1/6})$	$O(n^{-1/3})$	$\left(\frac{18\sigma^2_{C,i}}{\alpha'''(\theta)^2}\right)^{1/6}$
$\widehat{\alpha}_{F,ii}(\theta,h,n)$	$\frac{\sigma^2_{F,ii}}{nh}$	$\frac{1}{2}\alpha''(\theta)h$	$O(n^{-1/3})$	$O(n^{-1/3})$	$\left(\frac{2\sigma^2_{F,ii}}{\alpha''(\theta)^2}\right)^{1/3}$
$\widehat{\alpha}_{C,ii}(\theta,h,n)$	$\frac{\sigma^2_{C,ii}}{nh}$	$\frac{1}{6}\alpha'''(\theta)h^2$	$O(n^{-1/5})$	$O(n^{-2/5})$	$\left(\frac{9\sigma^2_{C,ii}}{4\alpha'''(\theta)^2}\right)^{1/5}$

Although the variance of the finite-difference estimators appear in the optimal values of h_*, it will be unlikely that they will be known in advance prior to estimating $\alpha'(\theta)$. They can however be estimated from preliminary runs and be combined with rough estimates of the derivatives of α to approximate h_*. That is, for $\widehat{\alpha}_{C,ii}(\theta, h, n)$,

$$\hat{h}_* = \left(\frac{9S^2_{C,ii}}{4\hat{\alpha}'''(\theta)^2}\right)^{1/5} \tag{2.28}$$

with

[10] For detailed literature on this see Glynn (1989), Fox and Glynn (1989), Zazanis (1987), and Frolov and Chentsov (1963).

[11] For more detail see Glasserman (2004:381-386).

$$S^2_{C,ii} = \frac{1}{4(n-1)} \sum_{i=1}^{n} \left(Y_i(\theta+h) - Y_i(\theta-h) - (2h)\widehat{\alpha}_{C,ii}(\theta,h,n)\right)^2 \tag{2.29}$$

and

$$\widehat{\alpha'''}(\theta,h,n) = \sum_{i=1}^{n} \left(\frac{Y_i(\theta+3h) - 3Y_i(\theta+h) + 3Y_i(\theta-h) - Y_i(\theta-3h)}{(2h)^3} \right). \tag{2.30}$$

As stated in the previous section, $\widehat{\alpha}_{C,ii}(\theta,h,n)$ will be the best method for estimating a sensitivity. To optimise this, one would have to calculate h_* given in Table 2.1. Estimation of second derivatives seems to be much more difficult when trying to minimise the bias and variance (see Glasserman, 2004:384-385). The next section provides a much more accurate and unbiased method of estimating the sensitivities.

2.4.2 Pathwise Derivative Estimates

This section develops alternatives to finite-difference methods that estimate derivatives directly, without simulating at multiple parameter values. They do so by taking advantage of additional information about the dynamics and parameter dependence of a simulated process.

By using the same notation as in the previous section, the price of a derivative that depends on parameter θ is $\alpha(\theta)$, where $\alpha(\theta) = E[H(\theta)]$, with $H(\theta)$ the discounted payoff function. Now assuming the derivative of an expectation is equal to the expectation of the derivative, one can calculate

$$\alpha'(\theta) = \frac{\mathrm{d}\alpha(\theta)}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta} E[H(\theta)] = E\left[\frac{\mathrm{d}}{\mathrm{d}\theta} H(\theta)\right], \tag{2.31}$$

i.e. if the interchange of differentiation and expectation is justified, then Equation 2.31 can be used to construct an unbiased estimator of $\alpha'(\theta)$.[12] Glasserman (2004:388) gives an example of how to apply this theory to calculate the delta of a vanilla European option. This example contains important theory that will be used when applying this method to estimate the sensitivities of an ERBO.

[12] Glasserman (2004:387) provides a precise definition of this (this is beyond the scope of this research, and only the results will be used).

Let $H(\theta) = e^{-rT}\max\{\Psi_T(\theta) - K, 0\} = e^{-rT}[\Psi_T(\theta) - K]^+$ be the payoff from a call option with all symbols with their usual meanings, and $\Psi_T(\theta)$ the value of the underlying at time T, depending on all input parameters θ except K. Applying the chain rule for differentiation, one obtains

$$\frac{\mathrm{d}H(\theta)}{\mathrm{d}\theta} = \frac{\mathrm{d}H(\theta)}{\mathrm{d}\Psi_T(\theta)}\frac{\mathrm{d}\Psi_T(\theta)}{\mathrm{d}\theta}. \tag{2.32}$$

The following result can be used to determine the first of the two factors:

$$\frac{\mathrm{d}}{\mathrm{d}\Psi_T}\max\{0, \Psi_T - K\} = \begin{cases} 0, & \Psi_T < K \\ 1, & \Psi_T > K. \end{cases} \tag{2.33}$$

This derivative fails to exist at $\Psi_T = K$, but because the event $\Psi_T(\theta) = K$ has probability 0, H is almost surely differentiable with respect to $\Psi_T(\theta)$ and has derivative

$$\frac{\mathrm{d}H(\theta)}{\mathrm{d}\Psi_T(\theta)} = e^{-rT}\mathbb{1}_{\{\Psi_T(\theta) > K\}}. \tag{2.34}$$

For the second factor in Equation 2.32, one can simply calculate it from the function $\Psi_T(\theta)$. This implies that the pathwise estimator is

$$\frac{\mathrm{d}H(\theta)}{\mathrm{d}\theta} = e^{-rT}\mathbb{1}_{\{\Psi_T(\theta) > K\}}\frac{\mathrm{d}\Psi_T(\theta)}{\mathrm{d}\theta}, \tag{2.35}$$

which can easily be computed mostly from the already simulated $\Psi_T(\theta)$.

Note also that

$$\frac{\mathrm{d}}{\mathrm{d}\Psi_T}\max(0, K - \Psi_T) = \begin{cases} 0, & \Psi_T > K \\ -1, & \Psi_T < K. \end{cases} \tag{2.36}$$

According to Glasserman (2004:418-419) extensive numerical evidence accumulated across many models and applications indicates that the pathwise method, when applicable, provides the best estimates of sensitivities. Compared with finite-difference methods, pathwise estimates require less computing time and they directly estimate derivatives rather than differences. Both of these methods will be applied in the results chapter on estimating sensitivities.

2.5 OBJECTIVES

In the chapters that follow the literature discussed in this chapter will be applied to the ERBPO, comparing different methods to find the most efficient method.

The next chapter will determine the price of the ERBPO with the help of MC simulation in a simplistic way; it will then be improved mathematically so that the paths that need to be calculated using this method can be refined to a much simpler method. Chapter 4 focuses on how this refined method can be improved using VRTs. All the methods discussed in the literature review will be applied and the results will be compared to find the most efficient method. Chapter 5 will again use the results of the next chapter, but will improve results using QMC methods. They will again be compared to find the most efficient method.

Chapter 6 applies the literature discussed to estimate sensitivities. This research will then be concluded with a summary of all the most efficient methods in the final chapter. Appendix D provides the final algorithm for the price and sensitivities of the ERBO. This algorithm was programmed in `R` and the code is given in Appendix E.

2.6 SUMMARY

The literature review provided the theory needed to calculate the price and sensitivities of an ERBO. Methods on how to determine the price of an option and methods on how to make these simulations more efficient (VRTs and QMC) were discussed in this chapter. Methods for estimating the sensitivities for an option using MC methods were also discussed. This chapter was concluded with a discussion of the objectives on how the literature review will be applied to the ERBO. The next four chapters give the results obtained from applying the literature to the ERBO problem.

CHAPTER 3

GENERAL MONTE CARLO

In this chapter an algorithm to estimate the price of an ERBPO using MC methods will be presented. The pricing, using MC methods, will be done in a simplistic way, whereafter this pricing method will be refined and improved using a simple mathematical proof. With this mathematical proof the process will be simplified and a formula will be found that can simulate the final value of the portfolio in one step. Simulations were done using both these methods and the results compared.

3.1 METHODOLOGY

Only the pricing of a put will be discussed here, but everything can easily be changed to price a call.

The ERBPO is an option on a portfolio, Π. This portfolio consists of $w_{i,j\tau}$ units of asset $i = 1, 2$ with price $S_{i,T}$ at time T. The number of units that are held in each asset, change every time a rebalance takes place. Equation 1.1 on page 4 provides a method for determining the number of units held in each asset at any time. The option has a strike price of K and therefore the payoff at time T for this put option is $\max\{K - \Pi_T, 0\}$, with $\Pi_T = w_{1,j\tau}S_{1,T} + w_{2,j\tau}S_{2,T}$ and j being the timestamp of the last rebalance before expiry.

3.1.1 Simplistic Monte Carlo

The simplistic MC method is the most intuitive method that can be used to price the ERBPO. This method uses normal MC pricing methods without considering the mathematics behind the simulation any further. This section provides a discussion of the algorithms used to price the ERBPO using the simplistic method.

3.1.1.1 Reasoning and Mathematics

Following an adapted version of Algorithm 2.1 on page 7 one can price the ERBPO as follows:

Algorithm 3.1 (Steps for simplistic MC simulation of an ERBPO)

1. *Sample random paths for* Π_T *in a risk-neutral world.*
2. *Calculate the payoff from this derivative at time* T *with* $\max\{K - \Pi_T, 0\}$.
3. *Discount the simulated payoff with the risk-free rate back to time 0.*
4. *Repeat steps 1 to 3,* n *times.*
5. *Calculate the average of the* n *values in step 4.*

All of the above steps can be performed easily, except for the value of Π_T, which needs to be discussed in more detail. Calculating the value of Π_T will be done as follows:

Algorithm 3.2 (Steps for simplistic MC simulation of Π_T)

1. *Simulate the paths followed by the two correlated underlying assets with jumps at times of rebalancing and at* T, *ie.* $S_{i,j\tau}$ *with* $j = 0, 1, ..., \lfloor T/\tau \rfloor$ *and* $S_{i,T}$ *with* $i = 1, 2$.
2. *Calculate the number of units held in each asset at each rebalancing time* $w_{i,j\tau}$ *with* i *and* j *the same meaning as above for* $j = 0, 1, ..., \lfloor T/\tau \rfloor$.
3. *The value for* Π_T *is* $\Pi_T = w_{1,\lfloor T/\tau \rfloor \tau} S_{1,T} + w_{2,\lfloor T/\tau \rfloor \tau} S_{2,T}$.

In the next section the programmable algorithm (PA) will be given.

3.1.1.2 Algorithm

If the two algorithms in the previous section are taken into consideration, together with Equation 1.1 on page 4, Equations 2.5 and 2.6 on page 10, and Theorem 2.2 on page 10, then the following PA can be constructed:

Algorithm 3.3 (PA for the simplistic MC pricing of an ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{P}$ *- the price of the ERBPO*
SE *- the SE of the estimate*
CI *- 95% CI*
Algorithm:
$\underline{P} \leftarrow \underline{0}$ *[vector of length n]*

if $(T \bmod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \bmod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
for $l = 1$ *to* n
$w_i \leftarrow v_i \Pi_0 / S_{i,0}$ *for* $i = 1, 2$
$S_i \leftarrow S_{i,0}$ *for* $i = 1, 2$
$\Pi \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
$S_i \leftarrow S_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i \sqrt{\underline{\Delta t}[j]}\right)$ *for* $i = 1, 2$
$\Pi \leftarrow w_1 S_1 + w_2 S_2$
$w_i \leftarrow v_i \Pi / S_i$ *for* $i = 1, 2$
end for
$\underline{P}[l] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
end for
$\bar{P} \leftarrow \sum_{i=1}^{n} \underline{P}[i]/n$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{P}[i] - \bar{P})^2/(n-1)}$
$SE \leftarrow S_P/\sqrt{n}$
$CI \leftarrow [\bar{P} - 1.96 \times SE, \bar{P} + 1.96 \times SE]$ ■

The above general algorithm can be implemented in any appropriate mathematical or statistical computer software program. This can also be coded in `R` (R Development Core Team, 2010). In the next section, it will be showed that the above algorithm can be refined by using some mathematical techniques. It will be proved that the price of the ERBO is independent of the initial prices of the two underlying assets. Figure 3.1 on page 36 provides a graphical representation of Algorithm 3.3.

3.1.2 Refined Monte Carlo

This section will take the algorithm that is used in the simplistic approach and mathematically change it so that it can be simulated a lot faster. It will also be shown that the value of the portfolio does not, at any time, depend on the initial values of the stock prices, but only on Π_0. The algorithms will be given and this method will then be compared with the simplistic approach in the results section of this chapter.

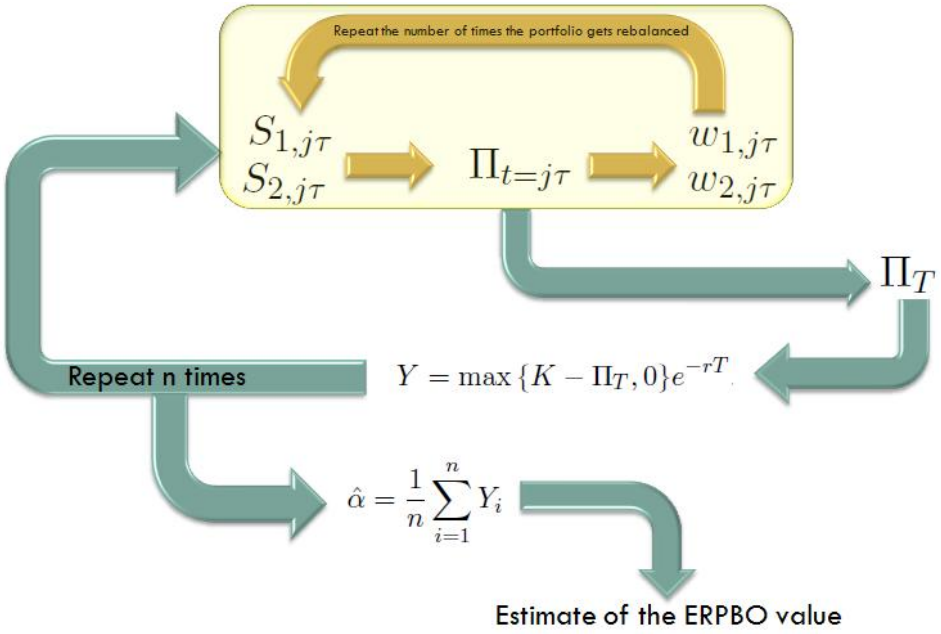

Figure 3.1: Graphical representation of Algorithm 3.3.

3.1.2.1 Reasoning and Mathematics

The value of the portfolio Π just before the j^{th} rebalancing can be calculated as follows:

$$\Pi^*_{j\tau} = \Pi_0 \prod_{l=1}^{j} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t + \sigma_i \epsilon_i \sqrt{\Delta t} \right) \right] \tag{3.1}$$

The above equation is derived in Appendix B. By using this to estimate the value of the portfolio at time T one can simply adjust the algorithm in the previous section so that the computational time drops considerably (see the results section). Also note that the above equation is totally independent of $S_{i,0}$ for $i = 1, 2$. If $T \bmod \tau \neq 0$, the above formula has to be adjusted to find an estimate of the value of the portfolio at time T, with $\underline{\Delta t}$ a vector of length $\lceil T/\tau \rceil$ so that $\underline{\Delta t} = [\tau, \tau, ..., \tau, T \bmod \tau]$:

$$\Pi_T = \Pi_0 \prod_{l=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t_l + \sigma_i \epsilon_i \sqrt{\Delta t_l} \right) \right]. \tag{3.2}$$

In the next section the algorithm that can be programmed in any mathematical/statistical program to calculate the price using this method, will be given.

3.1.2.2 Algorithm

Algorithm 3.4 (PA for the refined MC pricing of an ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{P}$ - *the price of the ERBPO*
SE - *the SE of the estimate*
CI - *95% CI*
Algorithm:
$\underline{P} \leftarrow \underline{0}$ *[vector of length n]*
if $(T \mod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \bmod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
for $l = 1$ *to* n
$\Pi \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i \sqrt{\underline{\Delta t}[j]}\right)$
end for
$\underline{P}[l] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
end for
$\bar{P} \leftarrow \sum_{i=1}^{n} \underline{P}[i]/n$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{P}[i] - \bar{P})^2/(n-1)}$
$SE \leftarrow S_P/\sqrt{n}$
$CI \leftarrow [\bar{P} - 1.96 \times SE, \bar{P} + 1.96 \times SE]$ ■

Figure 3.2 on page 38 provides a graphical representation of the above algorithm.

3.2 RESULTS

In this section the Simplistic MC approach is compared to the Refined MC approach. The simulations were done over different values of ρ, Π_0 and n. The other parameters were held fixed at: $\tau = 1$, $v_1 = v_2 = 0.5$, $T = 10$, $S_{1,0} = 15$, $S_{2,0} = 20$, $r = 0.03$, $\sigma_1 = \sigma_2 = 0.3$ and $K = 1\ 000$. The initial seed for the random number generator was also fixed so that results from different methods could be compared. As seen in the two results tables that follow, the Price and SE are exactly the same with the only difference being computing time.

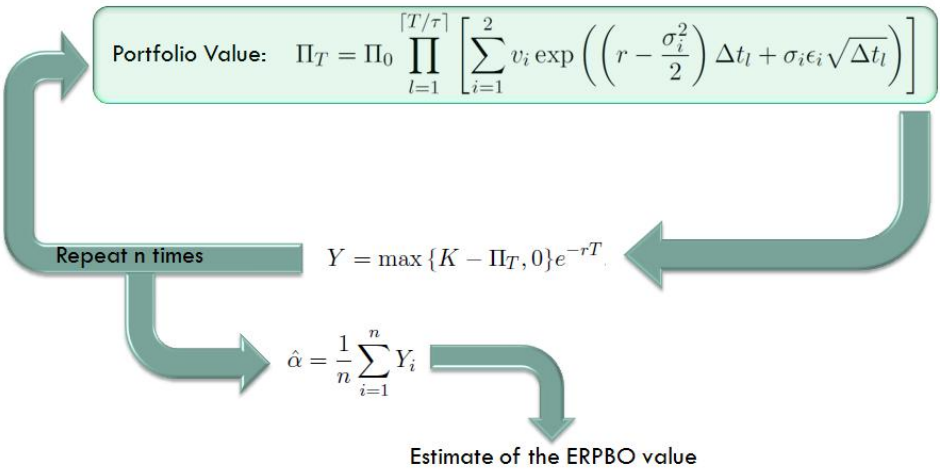

Figure 3.2: Graphical representation of Algorithm 3.4.

The computing time is considerably less for the refined method (Table 3.2 on page 40) than for the simplistic method (see Table 3.1 on page 39). It is interesting to note that in nominal terms the computing times are considerably reduced, for example when $n = 500\ 000$ the simplistic approach took ± 340 seconds, where the refined method only took ± 8 seconds - a reduction of $\pm 5\frac{1}{2}$ minutes. In real terms, the refined method only took $\pm 2\%$ of the computing time of the simplistic method - a $\pm 98\%$ reduction.

A graphical representation of these two tables of results can be found on page 41 as Figure 3.3.

3.3 CONCLUSION

In this chapter it was shown that the refined MC pricing method is more efficient than the simplistic approach. This was done by comparing different simulations of the two methods with the same seeds for the random number generator. Both methods obtained exactly the same price and SE, but the refined method was superior in terms of computing times. Therefore, the refined method will be used in the following chapters in an attempt to make it more efficient. In the next chapter this is done using VRTs.

Table 3.1: Price, SE and Computing time (in seconds) for the simplistic MC pricing of an ERBPO with $\tau = 1$, $v_1 = v_2 = 0.5$, $T = 10$, $S_{1,0} = 15$, $S_{2,0} = 20$, $r = 0.03$, $\sigma_1 = \sigma_2 = 0.3$, $K = 1\ 000$, and different values of ρ, Π_0, and n.

Problem	ρ	Π_0	n	Price	SE	Time (s)
1	-0.50	500	5 000	275.61	2.54	3.41
1			50 000	278.28	0.81	33.89
1			500 000	277.53	0.26	338.53
2		1 000	5 000	68.36	1.72	3.36
2			50 000	70.76	0.54	34.13
2			500 000	70.22	0.17	341.20
3		1 500	5 000	18.22	0.88	3.41
3			50 000	17.69	0.27	34.17
3			500 000	17.62	0.09	339.64
4	0	500	5 000	308.89	2.97	3.45
4			50 000	311.85	0.94	34.25
4			500 000	310.87	0.30	344.42
5		1 000	5 000	120.72	2.43	3.49
5			50 000	123.44	0.77	35.20
5			500 000	122.69	0.24	340.32
6		1 500	5 000	53.50	1.69	3.36
6			50 000	53.60	0.53	34.11
6			500 000	53.07	0.17	339.28
7	0.50	500	5 000	337.99	3.22	3.40
7			50 000	340.38	1.02	34.07
7			500 000	339.34	0.32	338.89
8		1 000	5 000	163.96	2.88	3.38
8			50 000	166.18	0.91	34.05
8			500 000	165.60	0.29	338.34
9		1 500	5 000	89.69	2.28	3.37
9			50 000	90.04	0.72	33.94
9			500 000	89.47	0.23	338.60

Table 3.2: Price, SE and Computing time (in seconds) for the refined MC pricing of an ERBPO with $\tau = 1$, $v_1 = v_2 = 0.5$, $T = 10$, $S_{1,0} = 15$, $S_{2,0} = 20$, $r = 0.03$, $\sigma_1 = \sigma_2 = 0.3$, $K = 1\ 000$, and different values of ρ, Π_0, and n.

Problem	ρ	Π_0	n	Price	SE	Time (s)
1	-0.5	500	5 000	275.61	2.54	0.06
1			50 000	278.28	0.81	0.68
1			500 000	277.53	0.26	8.00
2		1 000	5 000	68.36	1.72	0.15
2			50 000	70.76	0.54	0.67
2			500 000	70.22	0.17	7.89
3		1 500	5 000	18.22	0.88	0.07
3			50 000	17.69	0.27	0.78
3			500 000	17.62	0.09	8.00
4	0	500	5 000	308.89	2.97	0.06
4			50 000	311.85	0.94	0.77
4			500 000	310.87	0.30	7.95
5		1 000	5 000	120.72	2.43	0.16
5			50 000	123.44	0.77	0.67
5			500 000	122.69	0.24	7.79
6		1 500	5 000	53.50	1.69	0.16
6			50 000	53.60	0.53	0.69
6			500 000	53.07	0.17	7.87
7	0.5	500	5 000	337.99	3.22	0.08
7			50 000	340.38	1.02	0.77
7			500 000	339.34	0.32	7.97
8		1 000	5 000	163.96	2.88	0.14
8			50 000	166.18	0.91	0.68
8			500 000	165.60	0.29	7.89
9		1 500	5 000	89.69	2.28	0.08
9			50 000	90.04	0.72	0.78
9			500 000	89.47	0.23	7.97

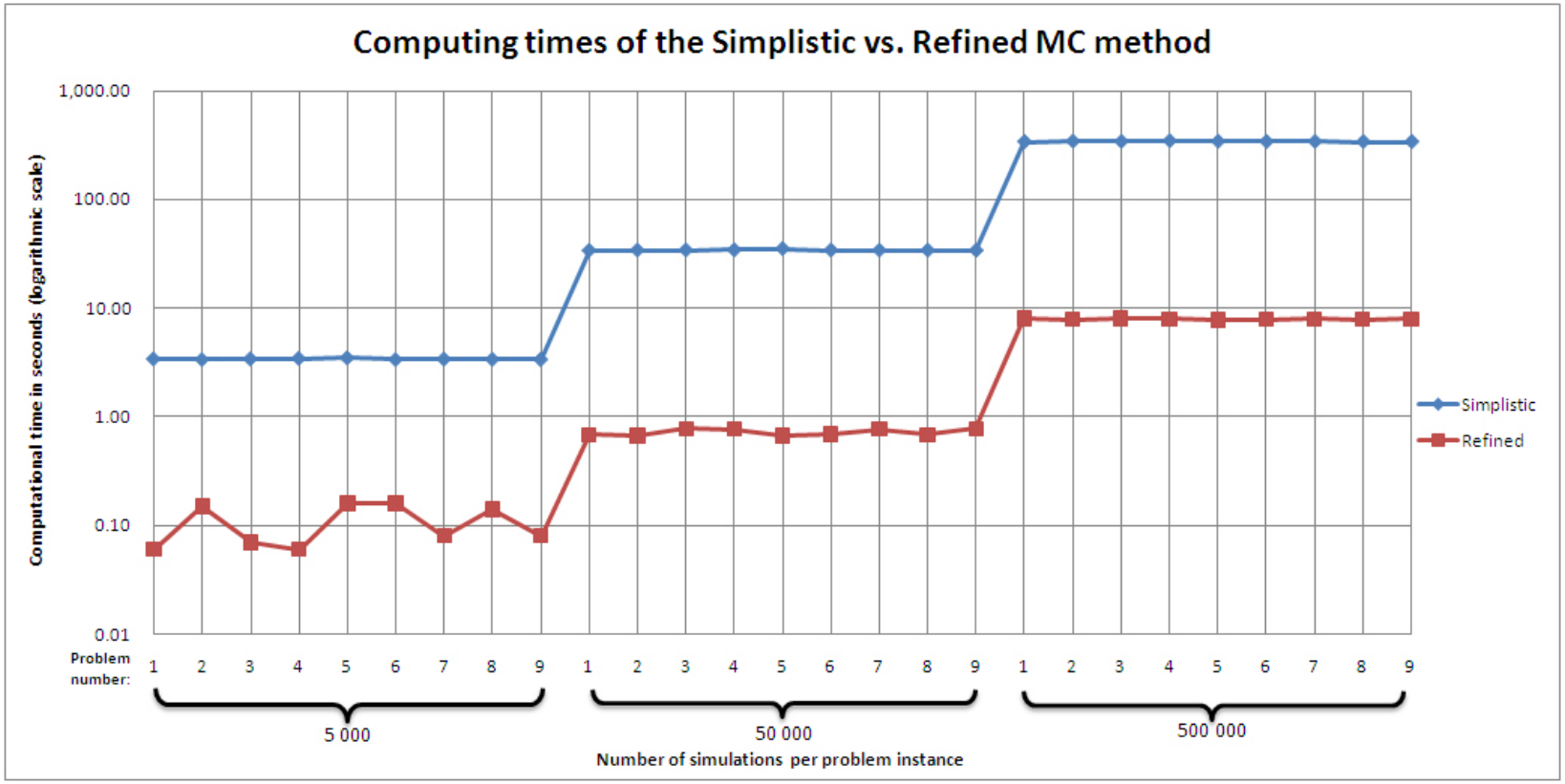

Figure 3.3: Graphical representation of computing times of the Simplistic vs. Refined MC methods.

CHAPTER 4

VARIANCE REDUCTION TECHNIQUES

This chapter applies three different VRTs to the refined MC simulation of the ERBPO. All the methodology behind each VRT in terms of the ERBPO will be supplied, followed by a general algorithm that needs to be implemented in order to produce a PA. The chapter concludes with a comparison of results, facilitating a choice with regards to the best method.

4.1 METHODOLOGY

In MC simulation $\lambda = E[X]$ is estimated by generating a sample $\{X_i : i = 1, ..., n\}$ and then determining $\widehat{\lambda}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$. Furthermore, the SE of the estimator is $\frac{\sigma}{\sqrt{n}}$ with σ^2 the variance of X. Note that there are two elements affecting the SE, namely $\sqrt{n}$ and σ. The first element can easily be interpreted: the more simulations that are done, the smaller the SE will become, and the more accurate the estimate will be. The other element is the square root of the variance of the simulated variable X. Therefore, to make the SE smaller, the variance of X should be reduced.

Due to the randomness of the sample generated to estimate λ and using the CLT (Theorem 2.3) a $(1-\beta)100\%$ CI can be constructed (see Definition 2.3), viz.

$$\left(\hat{\lambda}(n) - z_{\beta/2}\frac{\hat{\sigma}}{\sqrt{n}}; \hat{\lambda}(n) + z_{\beta/2}\frac{\hat{\sigma}}{\sqrt{n}}\right). \tag{4.1}$$

Therefore, if the SE can be reduced, then a smaller CI can be obtained. This chapter aims to reduce the size of s. The best method will be chosen on the basis of computational time and size of SE. That is, given a fixed amount of time, which method rendered the smallest SE and therefore the smallest CI?

The methods that will be discussed are AVs, CVs and LHS.

4.1.1 Antithetic Variables

Variance reduction by using AVs, discussed in Section 2.2.1, will be applied to the refined MC simulation of the ERBPO. The general algorithm will be given here, after which the PA will be given. AVs as a VRT will be compared with other methods in the last section.

4.1.1.1 Reasoning and Mathematics

Simulation of a path of the value of the portfolio depends on uniform variables simulated from the $d = 2 \times \lceil T/\tau \rceil$ dimensional unit hypercube $[0,1]^d$. Therefore let $\Pi_A = H(U_1, ..., U_d)$ and $\Pi_B = H(V_1, ..., V_d) = H(1-U_1, ..., 1-U_d)$ where H is the function that uses the d random variables to simulate a path for the final value of the portfolio. These are then used to estimate the price of the option, i.e.:

$$\alpha = E\left[\left(\frac{\max\{K-\Pi_A, 0\} + \max\{K-\Pi_B, 0\}}{2}\right) e^{-rT}\right]. \tag{4.2}$$

The above equation can be used to construct an unbiased estimator of α, and according to the CLT for large n, a CI as in Equation 4.1 can be constructed for the price.

4.1.1.2 Algorithm

Algorithm 4.1 (PA for the refined MC pricing of an ERBPO with AV)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{P}$ - *the price of the ERBPO*
SE - *the SE of the estimate*
CI - *95% CI*
Algorithm:
$\underline{P} \leftarrow \underline{0}$ *[vector of length n]*
if $(T \mod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \mod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
for $l = 1$ *to* n
$\Pi_A \leftarrow \Pi_0$

$\Pi_B \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate $\epsilon_{1,A}$ *and* $\epsilon_{2,B}$ *using the IPIT and Equations 2.7 & 2.8*
from $U_{1,A}$ *and* $U_{2,A}$ *with* $[U_{1,A}, U_{2,A}] \sim U(0,1)^2$
generate $\epsilon_{1,B}$ *and* $\epsilon_{2,B}$ *using the IPIT and Equations 2.7 & 2.8*
from $(1 - U_{1,A})$ *and* $(1 - U_{2,A})$
$\Pi_A \leftarrow \Pi_A \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_{i,A}\sqrt{\underline{\Delta t}[j]}\right)$
$\Pi_B \leftarrow \Pi_B \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_{i,B}\sqrt{\underline{\Delta t}[j]}\right)$
end for
$\underline{P}[l] \leftarrow (\max\{K - \Pi_A, 0\} + \max\{K - \Pi_B, 0\})e^{-rT}/2$
end for
$\bar{P} \leftarrow \sum_{i=1}^{n} \underline{P}[i]/n$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n}(\underline{P}[i] - \bar{P})^2/(n-1)}$
$SE \leftarrow S_P/\sqrt{n}$
$CI \leftarrow [\bar{P} - 1.96 \times SE, \bar{P} + 1.96 \times SE]$ ■

4.1.2 Control Variates

The next VRT that will be applied to the refined MC method is the implementation of a CV. The methodology behind CVs was discussed in the literature review and although the theory remains the same, the CV that will be implemented with the ERBPO will change somewhat. In the results section of this chapter it will be shown that CVs as a VRT can reduce the SE significantly.

4.1.2.1 Reasoning and Mathematics

The simplest CV that can be implemented is the price of a vanilla call or put. Usually the reason for this is the fact that the price of the underlying stock is already simulated with the simulation of the main option (usually exotic). Unfortunately, due to the nature of the refined method, the two underlying stocks are not simulated individually. Note that in the simplistic method this is done, but as shown earlier this method's computational time is substantially longer.

By taking the transformed simulated uniform numbers from the d dimensional unit hypercube, $\underline{U} \sim Unif(0,1)^d$, one can easily simulate paths for the individual assets (with the cost of extra computational time). This can be done as follows:

If

$$\Pi_T = \Pi_0 \prod_{l=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\Delta t_l + \sigma_i \epsilon_i \sqrt{\Delta t_l}\right)\right] \tag{4.3}$$

let

$$\Theta_{i,l} = \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\Delta t_l + \sigma_i \epsilon_i \sqrt{\Delta t_l}\right), \quad i = 1, 2 \tag{4.4}$$

so that

$$\Pi_T = \Pi_0 \prod_{l=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \Theta_{i,l}\right]. \tag{4.5}$$

Using $\Theta_{i,l}$ the prices of two normal assets with initial prices, Π_0, can easily be computed with

$$S_{i,T} = \Pi_0 \prod_{l=1}^{\lceil T/\tau \rceil} \Theta_{i,l}, \quad i = 1, 2 \tag{4.6}$$

so that $S_{i,T}$ for $i = 1, 2$ can be used in the simulation of the prices of two vanilla calls/puts.

The final CV is the sum of two vanilla calls/puts (depending on whether simulating an ERBCO or an ERBPO) on asset 1 and asset 2 with strike price K. Hence, Algorithm 2.4 is changed as follows:

Algorithm 4.2 (Steps for MC simulation with CVs)

1. *For $i = 1, ..., n_1$ simulate n_1 independent paths and obtain a sample of Π_T and $S_{i,T}$ values. With these calculate and obtain a sample of discounted payoffs from the option C_i (the ERBO) as well as the sum of the two vanilla options ($O_i = O_{1,i} + O_{2,i}$).*
2. *Calculate c^* with the help of Equation 2.18.*
3. *Repeat simulation as normal and simulate n_2 samples of Π_T and $S_{i,T}$ and calculate the discounted payoffs from it.*
4. *Calculate $C_{CV} = C_i + c^*(O_i - E[O])$, where $E[O]$ is the sum of the theoretical prices of the two vanilla options using the BS pricing formula (see Appendix A.4).* ■

The PA is given in the next section, after which this method will be compared to the other VRTs in the results section.

4.1.2.2 Algorithm

Algorithm 4.3 (PA for the refined MC pricing of an ERBPO using CVs)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{P}_{CV}$ - *the price of the ERBPO*
SE - *the SE of the estimate*
CI - *95% CI*
Algorithm:
$n_1 \leftarrow \lfloor 0.05 \times n \rfloor$
$n_2 \leftarrow n - n_1$
$\underline{P} \leftarrow \underline{0}$ *[vector of length* n_1*]*
$\underline{CV} \leftarrow \underline{0}$ *[vector of length* n_1*]*
if $(T \mod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \mod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
for $l = 1$ *to* n_1
$\Pi \leftarrow \Pi_0$
$CV_1 \leftarrow \Pi_0$
$CV_2 \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
$\Theta_1 \leftarrow \exp\left(\left(r - \frac{\sigma_1^2}{2}\right)\underline{\Delta t}[j] + \sigma_1 \epsilon_1[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
$\Theta_2 \leftarrow \exp\left(\left(r - \frac{\sigma_2^2}{2}\right)\underline{\Delta t}[j] + \sigma_2 \epsilon_2[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i \Theta_i$
$CV_1 \leftarrow CV_1 \times \Theta_1$
$CV_2 \leftarrow CV_2 \times \Theta_2$
end for
$\underline{P}[l] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
$\underline{CV}[l] \leftarrow \max\{K - CV_1, 0\} e^{-rT} + \max\{K - CV_2, 0\} e^{-rT}$
end for
$\bar{P} \leftarrow \sum_{i=1}^{n_1} \underline{P}[i]/n_1$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n_1} (\underline{P}[i] - \bar{P})^2/(n_1 - 1)}$
$\bar{CV} \leftarrow BS_p(S_0 = \Pi_0, T = T, r = r, \sigma = \sigma_1, K = K) +$
$BS_p(S_0 = \Pi_0, T = T, r = r, \sigma = \sigma_2, K = K)$
$\sigma_{P,CV} \leftarrow \sum_{i=1}^{n_1} (\underline{P}[i] - \bar{P})(\underline{CV}[i] - \bar{CV})/n_1$
$c^* \leftarrow -\sigma_{P,CV}/S_P^2$
$\underline{P} \leftarrow \underline{0}$ *[vector of length* n_2*]*

$\underline{P}_{CV} \leftarrow \underline{0}$ *[vector of length* n_2*]*
$\underline{CV} \leftarrow \underline{0}$ *[vector of length* n_2*]*
for $l = 1$ *to* n_2
  $\Pi \leftarrow \Pi_0$
  $CV_1 \leftarrow \Pi_0$
  $CV_2 \leftarrow \Pi_0$
  for $j = 1$ *to* $\lceil T/\tau \rceil$
    generate ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
    $\Theta_1 \leftarrow \exp\left(\left(r - \frac{\sigma_1^2}{2}\right)\underline{\Delta t}[j] + \sigma_1\epsilon_1[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
    $\Theta_2 \leftarrow \exp\left(\left(r - \frac{\sigma_2^2}{2}\right)\underline{\Delta t}[j] + \sigma_2\epsilon_2[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
    $\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i\Theta_i$
    $CV_1 \leftarrow CV_1 \times \Theta_1$
    $CV_2 \leftarrow CV_2 \times \Theta_2$
  end
  $\underline{P}[l] \leftarrow \max\{K - \Pi, 0\}e^{-rT}$
  $\underline{CV}[l] \leftarrow \max\{K - CV_1, 0\}e^{-rT} + \max\{K - CV_2, 0\}e^{-rT}$
  $\underline{P}_{CV}[l] \leftarrow \underline{P}[l] + c^*(\underline{CV}[l] - \bar{CV})$
end
$\bar{P}_{CV} \leftarrow \sum_{i=1}^{n_2} \underline{P}_{CV}[i]/n_2$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n_2}(\underline{P}_{CV}[i] - \bar{P}_{CV})^2/(n_2 - 1)}$
$SE \leftarrow S_P/\sqrt{n_2}$
$CI \leftarrow [\bar{P}_{CV} - 1.96 \times SE, \bar{P}_{CV} + 1.96 \times SE]$ ■

4.1.3 Latin Hypercube Sampling

LHS is the method of systematic sampling for higher dimensions. This method helps with the reduction of variance by sampling systematically (evenly spread out) throughout the unit hypercube. It is important to note that by generating these n samples from the unit hypercube, two things must be known beforehand. Firstly, the dimension of the hypercube (d) should be known, and secondly the number of samples (n) that needs to be taken from this hypercube. The generated values are then dependent on the n and d that are used as initial inputs. This poses a problem: the 'randomness' of the samples that are generated is compromised, and because of this, it is not possible to measure the reduction in variance.

4.1.3.1 Reasoning and Mathematics

If one generates n samples from the unit hypercube $(0,1)^d$ using LHS it might as well have been a realisation of n samples from the unit hypercube $(0,1)^d$ using normal pseudo-random numbers. The only difference is that the LHS samples are chosen evenly through the unit hypercube. Therefore,

by simply calculating the SE of an estimate, exactly the same results as for the normal MC method would be obtained. No variance reduction will be visible.

Glasserman (2004:242) suggests generating i.i.d. estimators $\widehat{\alpha}_1(B), ..., \widehat{\alpha}_m(B)$, each based on a LHS of size B. Then an asymptotically (as $m \to \infty$) valid $(1-\beta)100\%$ CI for α can be calculated as

$$\left(\frac{1}{m}\sum_{i=1}^{m}\widehat{\alpha}_i(B)\right) \pm z_{\beta/2}\frac{\widehat{\sigma}}{\sqrt{m}}, \tag{4.7}$$

with $\widehat{\sigma}$ the sample standard deviation of $\widehat{\alpha}_1(B), ..., \widehat{\alpha}_m(B)$. He argues that the only cost in this approach lies in foregoing the possibly greater variance reduction from generating a single LHS of size nB rather than n independent samples of size B.

Therefore the algorithm that will be used can be described as follows:

Algorithm 4.4 (Steps for the refined MC pricing of an ERBPO using LHS)

1. *Estimate the price of an ERBPO $m = n/B$ times using Algorithm 3.4, but with B repetitions in each simulation, and using LHS to generate the uniform numbers from the d-dimensional unit hypercube.*
2. *Use the estimates obtained in the previous step to calculate the average, this is the estimate of the price of the ERBPO.*
3. *Also use these to calculate the SE and to construct a CI.* ■

The algorithm given in the next section can be used to program this method into any mathematical software package. The SE obtained in the algorithm described above will be compared to the ones obtained using the other methods in the results section.

4.1.3.2 Algorithm

Algorithm 4.5 (PA for the refined MC pricing of an ERBPO using LHS)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{P}_F$ *- the price of the ERBPO*
SE *- the SE of the estimate*
CI *- 95% CI*
Algorithm:
$n_1 \leftarrow \lfloor 0.1n \rfloor$
$n_2 \leftarrow n - n_1$
$\underline{P} \leftarrow \underline{0}$ *[vector of length* n_2*]*

$\underline{P}_F \leftarrow \underline{0}$ *[vector of length* n_1*]*
if $(T \mod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \bmod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
for $l = 1$ *to* n_1
Generate a matrix U *containing* n_2 *samples from the unit hypercube* $(0,1)^d$ *generated by LHS, with* $d = 2 \times \lceil T/\tau \rceil$.
Split each sample into two vectors of length $\lceil T/\tau \rceil$*, thus* $\underline{U}_{1,i}$ *and* $\underline{U}_{2,i}$ *for* $i = 1, ..., n_2$
for $m = 1$ *to* n_2
$\Pi \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8 from* $\underline{U}_{1,m}[j]$ *and* $\underline{U}_{2,m}[j]$
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i \sqrt{\underline{\Delta t}[j]}\right)$
end for
$\underline{P}[m] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
end for
$\underline{P}_F[l] \leftarrow \sum_{i=1}^{n_2} \underline{P}[i]/n_2$
end for
$\bar{P}_F \leftarrow \sum_{i=1}^{n_1} \underline{P}_F[i]/n_1$
$S_{P_F} \leftarrow \sqrt{\sum_{i=1}^{n_1} (\underline{P}_F[i] - \bar{P}_F)^2/(n_1 - 1)}$
$SE \leftarrow S_{P_F}/\sqrt{n_1}$
$CI \leftarrow [\bar{P}_F - 1.96 \times SE, \bar{P}_F + 1.96 \times SE]$ ■

4.2 RESULTS

To determine the efficiency of VRTs, the various methods were compared with each other. In total 1 620 problem instances were constructed through all combinations of the following parameters: the correlation $\rho \in \{-1, -0.5, 0, 0.5, 1\}$; the initial value of the portfolio $\Pi_0 \in \{500, 1\ 000, 1\ 500\}$; the time between rebalancing $\tau \in \{0.25, 0.5, 1\}$; the proportion invested in the first asset $v_1 \in \{0.1, 0.5\}$; the time till maturity $T \in \{2, 10, 15\}$; the risk-free rate $r \in \{0.01, 0.03, 0.08\}$; and the standard deviation of the first asset $\sigma_1 \in \{0.05, 0.3\}$, $\sigma_2 = 0.3$ and $K = 1\ 000$.

The ERBPO was priced using the refined RP method, as well as the refined RP method with AV, CV and LHS as VRTs using all possible combinations of the above parameters. The number of repetitions for each of the 1 620 combinations were adjusted so that the computational time only

lasted approximately 1 second. That is, given 1 second, which method will yield relative to each other, the smallest SE and therefore the narrowest CI? Due to the extent of the results, only a summary can be provided here.

Table 4.1 counts the number of times a certain VRT performed the best (or outperformed others), i.e. produced the smallest SE in one second. The second column (Smallest) counts the number of times a VRT performed the best compared to the other methods. The NA (Not Available) row indicates the situation where there was no clear winner. The next five columns count the number of times a certain VRT outperformed the VRT that performed second best in terms of smallest SE. The margins by which the VRTs outperform the second best are 0.05, 0.1, 0.2, 0.5 and 0.75.

Table 4.1: Comparison of VRTs, with the number of times a certain VRT outperformed the second best VRT given one second computational time.

Method	Smallest	>0.05	>0.1	>0.2	>0.5	>0.75
Normal	43	0	0	0	0	0
AV	793	364	249	120	7	1
CV	548	470	398	250	46	8
LHS	196	37	12	10	0	0
NA	40	749	961	1 240	1 567	1 611
TOTAL	1 620	1 620	1 620	1 620	1 620	1 620

Initially, if only the 'Smallest' column is considered, AV outperformed all of the other VRTs. However, considering by how much AV reduces the variance with respect to other methods, it soon does not hold up to its initial reputation. The amount of times AV 'won' soon reduces much faster than that of CV when looking at by how much it outperformed the second most efficient VRT. For example, in 470 of the 1 620 instances the CV method produced an SE that was more than 0.05 smaller than the second best method, but AV only won in 364 of the 1 620 cases.

Figure 4.1 on page 52 provides a graphical representation of the above results.

4.3 CONCLUSION

CVs will be chosen as the better VRT. Although initially AVs performed the best it did not beat CVs as the margin to determine the best method increase. Therefore, when applying this VRT, the SE of the estimate should mostly be smaller than for other methods.

Further analysis can be done on how one can optimally choose which method works best for certain input parameters. One could also try to combine two or more of the VRTs and then compare that method with the 'traditional' methods. This, however, is beyond the scope of this research.

In the next chapter QMC methods will be used to price the options. This method differs somewhat from the VRTs due to the fact that it is totally deterministic and therefore no CI can be estimated, but by randomising some of the sequences, CIs can be estimated.

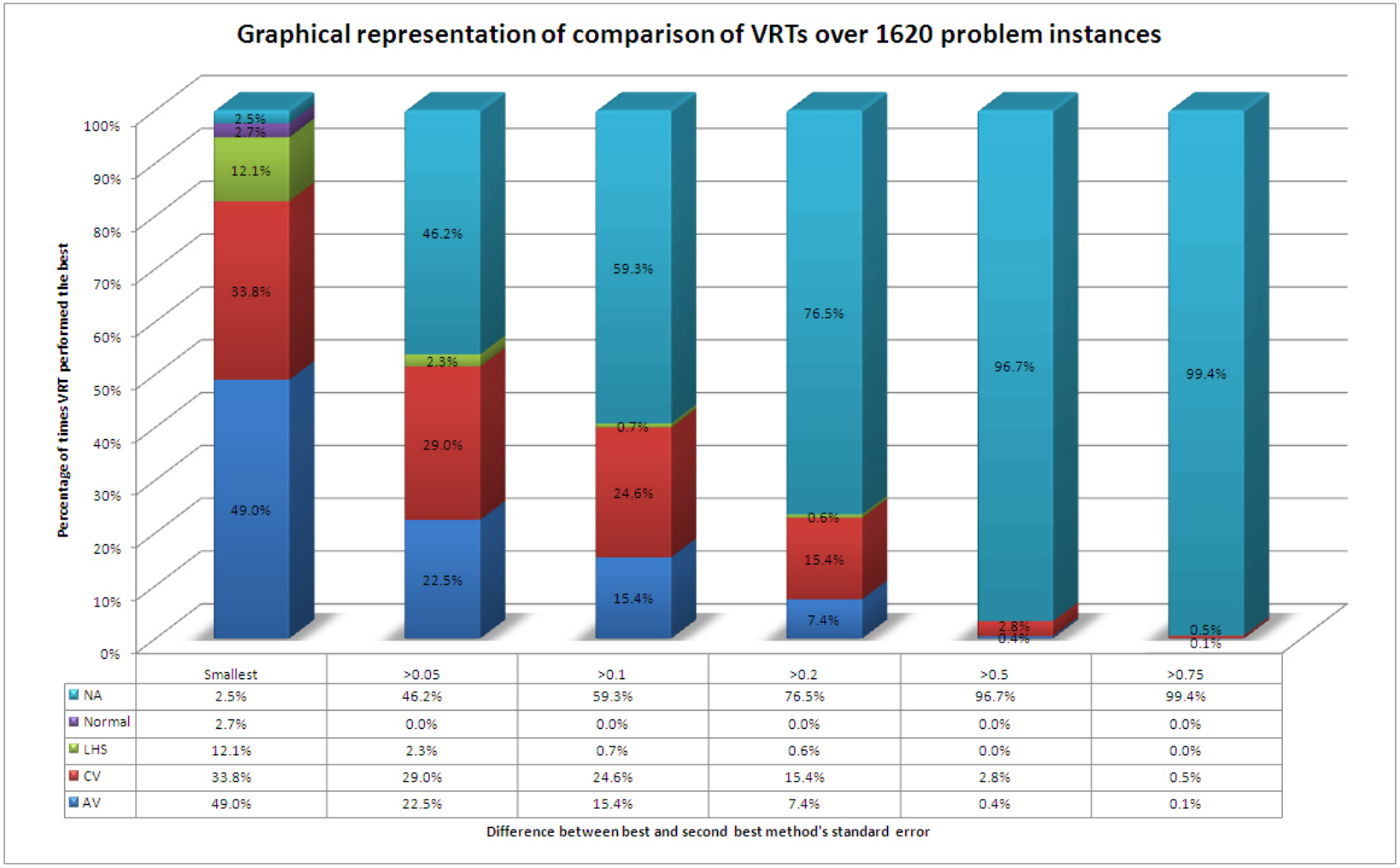

Figure 4.1: Graphical representation of comparison of VRTs over 1 620 problem instances.

CHAPTER 5

QUASI-MONTE CARLO TECHNIQUES

In this chapter the normal refined MC method and application of QMC techniques are considered. QMC techniques attempt to accelerate convergence of the method by using LDSs which are deterministically chosen sequences. The methodology behind the implementation of QMC in the ERBO problem is first discussed, after which an algorithm is provided that will be used when implementing QMC. The results section is divided into two subsections which compare the different LDSs.

5.1 METHODOLOGY

The theory behind QMC techniques was discussed in the literature review. The basic algorithm for pricing stays exactly the same, only the generation of the $\underline{U}$'s changes from random $Unif(0,1)^d$ values to those of the LDSs. The methodology on how this was implemented will be discussed briefly since all the code for generating the LDSs are readily available in `R` (R Development Core Team, 2010), and the randomised LDSs (RLDSs) can be constructed with ease as well.

The three LDSs that were implemented are the Halton, Faure and Sobol' sequences. The algorithm used for pricing the ERBPO was exactly the same as Algorithm 3.4 except LDSs were used as the $\underline{U} \sim Unif(0,1)^d$ instead of generating them with the help of a random number generator. Four RLDSs were also implemented, are all based on the SS as this (as will be shown in the result section) is the best performing sequence of the three normal LDSs discussed. The first RLDS was discussed in the literature review, where $\underline{U} \sim Unif(0,1)^d$ is generated with the random number generator and then added to each point of the SS modular 1 (Sobol' R). The next three RLDSs were not discussed in the literature review as the mathematics behind them are beyond the scope of this research. These scrambling methods are built into `R`, and include: Owen type scrambling (Sobol' R1); Faure-Tezuka type scrambling (Sobol' R2); and both Owen and Faure-Tezuka type scrambling (Sobol' R3).

Algorithm 5.1 (PA for the refined MC pricing of an ERBPO using QMC)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{P}$ - *the price of the ERBPO*
SE - *the SE of the estimate*
CI - *95% CI*
Algorithm:
$\underline{P} \leftarrow \underline{0}$ *[vector of length n]*
if $(T \mod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \mod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
generate a $2\lceil T/\tau \rceil \times n$ *matrix containing an LDS*
for $l = 1$ *to* n
$\Pi \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate ϵ_1 *and* ϵ_2 *using the IPIT (Definition 2.1) and Equations 2.7 & 2.8 with the* l^{th} *column from the LDS matrix*
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i \sqrt{\underline{\Delta t}[j]}\right)$
end for
$\underline{P}[l] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
end for
$\bar{P} \leftarrow \sum_{i=1}^{n} \underline{P}[i]/n$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{P}[i] - \bar{P})^2/(n-1)}$
$SE \leftarrow S_P/\sqrt{n}$
$CI \leftarrow [\bar{P} - 1.96 \times SE, \bar{P} + 1.96 \times SE]$ ■

Note however, that the CI is only valid if using an RLDS.

5.2 RESULTS

The results section will be divided into two sections: the first will compare the different LDSs and RLDSs over different inputs for the ERBPO, with the dimension kept constant at 4 and the number of points used to estimate the price as the factor over which they will be compared; the second will compare the different LDSs and RLDSs over different inputs for the ERBPO, with the number of points used to estimate the price kept constant and the dimension being the factor over which the sequences will be compared.

This section follows the same comparison of the sequences as in Glasserman (2004:324-328). Therefore the first method will use the RMSE to compare the different values for the number of points used, and the RRMSE will be used to compare the different values for the dimension.

5.2.1 Constant dimensionality

The first results will be obtained with the following equation for the RSME,

$$RMSE(n) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \alpha_i)^2}, \tag{5.1}$$

with m ERBPO problems with true values $\alpha_1, ..., \alpha_m$ and n-point approximations $\widehat{\alpha}_1(n), ..., \widehat{\alpha}_m(n)$. Unfortunately the true values are not known, and will have to be estimated with MC. That is, let the true values be denoted by α_i and be estimated by $\widehat{\alpha}_{MC,i}(N^*)$ with $N^* \to \infty$. The LLN implies that this will be the true value (given N^* is sufficiently large). The RMSE equation now changes to

$$\begin{aligned}
RMSE(n) &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \widehat{\alpha}_{MC,i}(N^*))^2} \\
&= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \alpha_i + \alpha_i - \widehat{\alpha}_{MC,i}(N^*))^2} \\
&= \sqrt{\underbrace{\frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \alpha_i)^2}_{\to 0} + \underbrace{\frac{1}{m}\sum_{i=1}^{m}(\alpha_i - \widehat{\alpha}_{MC,i}(N^*))^2}_{\to d_{m,N^*}} + \underbrace{\text{Crossproduct}}_{\to 0}} \\
&\to d_{m,N^*} \quad \text{as} \quad n \to \infty \qquad (5.2) \\
&\to 0 \quad \text{as} \quad N^* \to \infty. \qquad (5.3)
\end{aligned}$$

This implies that, in practice, the RMSE will always converge to a number d_{m,N^*} due to the fact that $N^* \to \infty$ is computationally impossible. Nevertheless, one can still compare the different methods on how fast they converge to this value. Figure 5.1 on page 56 provides this comparison for the different LDSs, together with some RLDSs. Note that α_i was estimated with $\widehat{\alpha}_{MC,i}(N^*)$ using the normal refined MC algorithm with $N^* = 10^7$. The other parameters were chosen as follows: $\rho \in \{-1, -0.5, 0, 0.5, 1\}$, $\Pi_0 \in \{500,\ 1\ 000,\ 1\ 500\}$, $v_1 \in \{0.1, 0.25, 0.5\}$, $r \in \{0.01, 0.03, 0.08\}$, $\sigma_1 \in \{0.05, 0.1, 0.3, 0.5\}$, K =1 000, $T = 10$, $\tau = 2$ and $\sigma_2 = 0.3$. The values for n were chosen carefully to optimally fill the unit hypercube; they are $n \in \{4^4, 4^5, 4^6, 4^7, 4^8, 4^9\}$.

Figure 5.1 displays the graph for the first part of the results for this section. From the graph it is clear that the best performing sequence is one of the randomised SSs (R1, R2 or R3) as they converge much more quickly to the value d_{m,N^*}. This gives two advantages: faster convergence, and because this is a randomised sequence, it is possible to construct a CI.

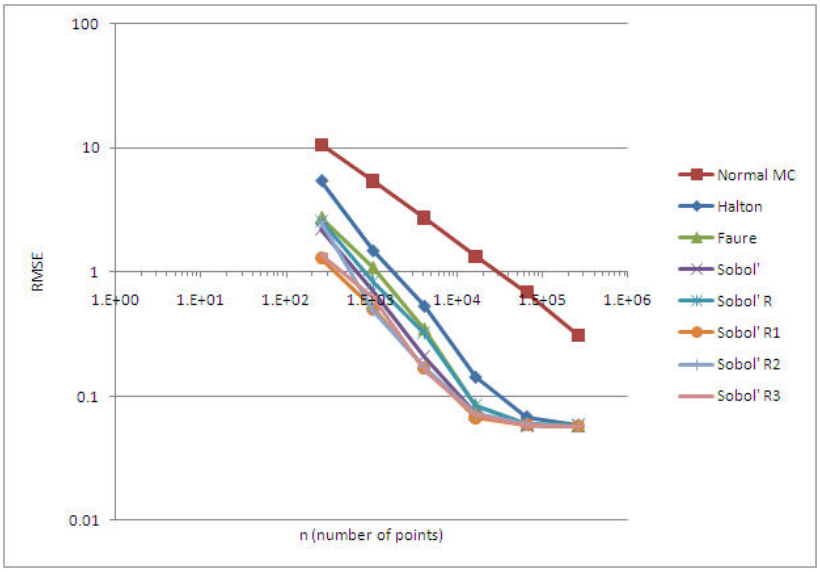

Figure 5.1: RMSE for different RLDSs over increasing values of n.

From this result it is interesting to note that, to obtain the same RMSE of 0.9 the normal MC simulation has to use $n = \pm 100\ 000$ simulations, compared to the Sobol' R3 method that only needs $n = \pm 1\ 000$. That is, only $\pm 1\%$ of the simulations are required to obtain the same accuracy - a $\pm 99\%$ reduction in the number of simulations.

5.2.2 Increasing dimensionality

The second part of this results section will state the results on how the LDSs and RLDSs performed over different values for the dimensions. As stated previously, Glasserman (2004:327) suggests using the RRMSE to compare the different sequences when considering increasing dimensionality. The formula is

$$RRMSE(n) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\widehat{\alpha}_i(n) - \alpha_i}{\alpha_i}\right)^2}, \tag{5.4}$$

with m ERBPO problems with true values $\alpha_1, ..., \alpha_m$ and n-point approximations $\widehat{\alpha}_1(n), ..., \widehat{\alpha}_m(n)$. Unfortunately the true values are not known, and will have to be estimated with MC. That is, let the true values be denoted by α_i and be estimated by $\widehat{\alpha}_{MC,i}(N^*)$ with $N^* \to \infty$.

For the results given in Figure 5.2, n was chosen as 5 120 and N^* as 900 000. The time between rebalancing was carefully chosen so that the dimension of the problems changes from 20, 40, 80 and to 200. The other parameters were chosen as follows: $\rho \in \{-1, -0.5, 0, 0.5, 1\}$, $\Pi_0 \in \{500,\ 1\ 000,\ 1\ 500\}$, $v_1 \in \{0.1, 0.25, 0.5\}$, $r \in \{0.01, 0.03, 0.08\}$, $\sigma_1 \in \{0.05, 0.1, 0.3, 0.5\}$, $K = 1\ 000$, $T = 10$, $\tau = 2$ and $\sigma_2 = 0.3$. The values for τ was chosen to obtain the desired dimension of the problems; they are $\tau \in \{2, 1, 0.25, 0.1\}$.

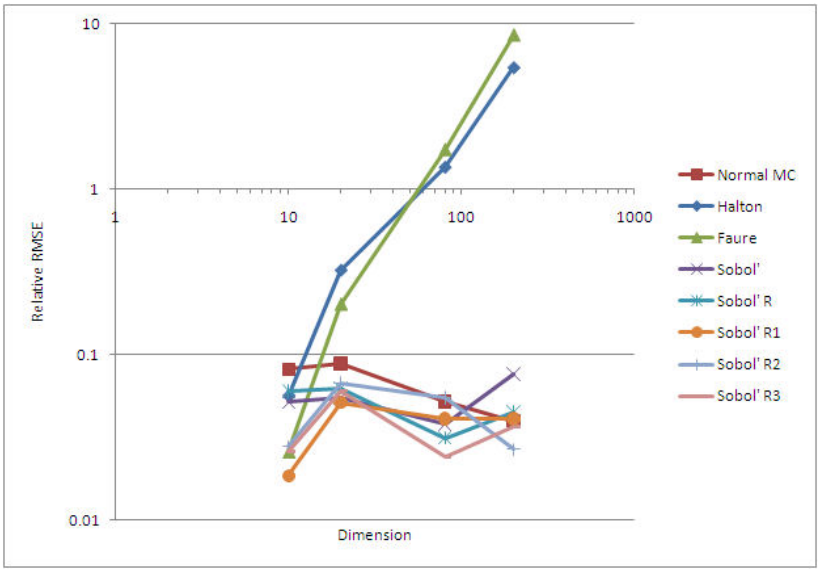

Figure 5.2: RRMSE for different RLDSs over increasing sizes of the dimension.

The FS and HS do not perform well at all due to the nature of the sequences (n has to be chosen carefully to obtain better results). But comparing the other LDSs, it is clear that the other methods all produce smaller RRMSEs than that of the normal MC method. The efficiency decreases, however, as the dimension increases - for example when $d = 10$ the Sobol' R1 method produced an RRMSE which was a lot smaller than the normal MC method, but when $d = 200$ they produced almost the same RRMSE.

For the sake of completeness of results, the RRMSE graph for the constant dimensionality case (Figure 5.3) and the RMSE for the increasing dimensionality case (Figure 5.4) are included on page 58. These graphs support the conclusion in the next section.

5.3 CONCLUSION

Taking all the results into consideration, it seems best to use the Sobol' sequence with Owen and Faure-Tezuka type scrambling, as they outperformed the others, obtaining not only faster convergence, but also the ability to construct a CI for an estimate. Figures 5.3 and 5.4 support the

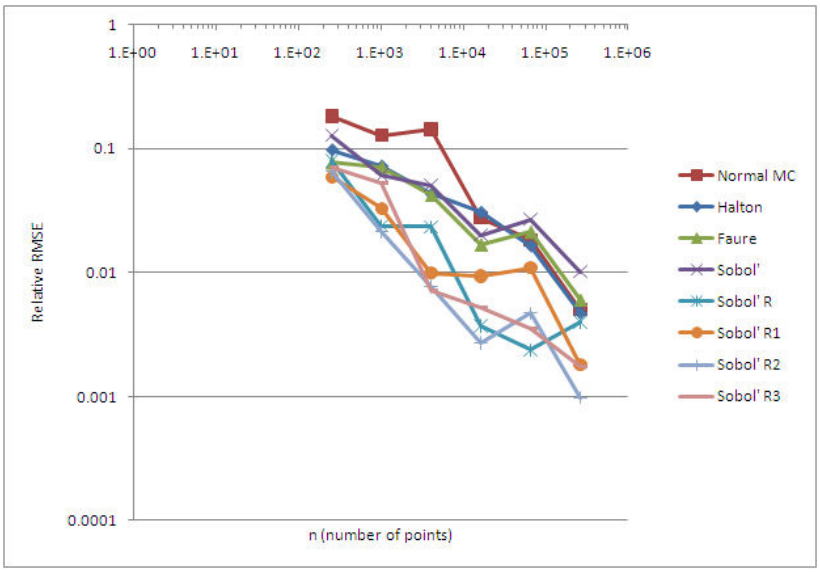

Figure 5.3: RRMSE for different RLDSs over increasing values of n.

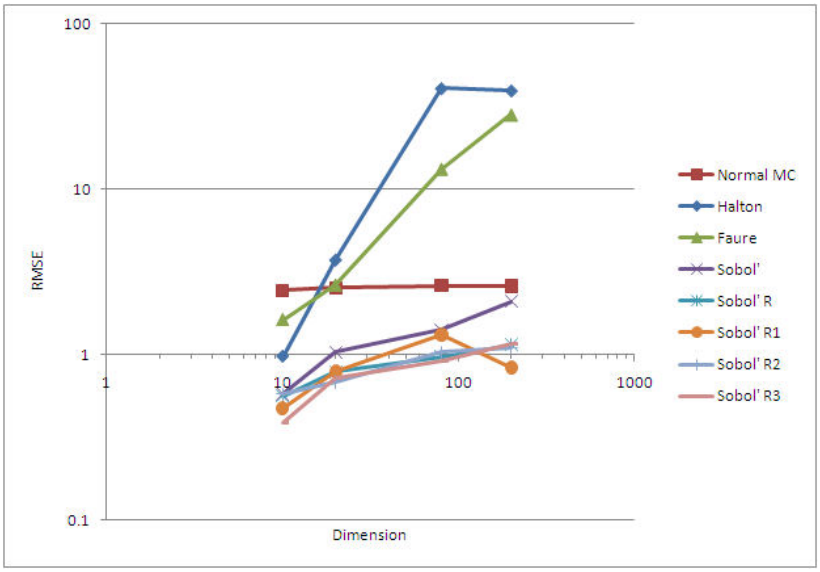

Figure 5.4: RMSE for different RLDSs over increasing sizes of the dimension.

conclusion, with the Sobol' sequence with Owen and Faure-Tezuka type scrambling (Sobol' R3) performing slightly better than the others.

In the next chapter methods on how to estimate sensitivities of the ERBPO are discussed.

CHAPTER 6

SENSITIVITY ANALYSIS

Sensitivity analysis is an important part of this research, as stated in both the introductory chapter and the literature review. In this chapter, two methods of estimating sensitivities for the ERBPO, FDAs and PDEs, are considered. The methodology on how these two methods will be applied to the ERBPO will be discussed in the next section, where algorithms for each of them will be provided as well. In the results section the different methods are discussed in terms of computing time and error of the estimates.

6.1 METHODOLOGY

The different methods will be applied to the refined MC pricing of the ERBPO (as given in Algorithm 3.4). For both methods the algorithm needs to be modified slightly for the computation of the sensitivities. The idea is to split up the computations so that there are more, already calculated, vectors and matrices of values to be used for further computations (this is particularly useful for PDE).

6.1.1 Finite-Difference Approximations

The method of approximating a sensitivity of an option using the FDA is exactly the same for all parameters. The best FDA method leads to the use of the central difference estimator, as discussed in Section 2.4.1.2:

$$\widehat{\alpha'}_{FD}(\theta, h, n) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i(\theta+h) - Y_i(\theta-h)}{2h}\right) \tag{6.1}$$

given in Equation 2.25, using common random numbers and choosing the optimal h_* as

$$\left(\frac{9\sigma_{FD}^2}{4\alpha'''(\theta)^2}\right)^{1/5}. \tag{6.2}$$

Then σ_{FD}^2 can be estimated with a sample variance

$$S_{FD}^2 = \frac{1}{4(n-1)}\sum_{i=1}^{n}\left(Y_i(\theta+h) - Y_i(\theta-h) - (2h)\widehat{\alpha'}_{FD}(\theta,h,n)\right)^2. \tag{6.3}$$

An estimate for $\alpha'''(\theta)$ is also required and the following can be used:

$$\widehat{\alpha'''}_{FD}(\theta,h,n) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i(\theta+3h) - 3Y_i(\theta+h) + 3Y_i(\theta-h) - Y_i(\theta-3h)}{(2h)^3}\right) \tag{6.4}$$

for some small h. Therefore, to apply FDA, preliminary runs are needed to estimate h_*, which is then used to estimate the sensitivity of the ERBPO with respect to different parameters. Only estimation of the sensitivity with respect to Π_0 will be given here, as the method for the other parameters is identical. The initial h is chosen as $10\%\theta$ (with θ the initial value of the parameter) and the number of simulations for the preliminary runs are chosen as 10% of the full number of simulations chosen as input, n. Note that the random number generator's seed needs to be fixed when calculating prices at different input parameters.

6.1.1.1 Algorithm

The general algorithm that needs to be applied is as follows:

Algorithm 6.1 (Steps for FDA of first derivative of θ)

1. *Generate* $10\% \times n \times \lceil T/\tau \rceil$ *values of* ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
2. *Use these random numbers to generate vectors containing* $10\%n$ *simulated discounted payoffs of the ERBO, that is* $Y_i(\theta + kh)$ *with* $k \in \{3, 1, -1, 3\}$
3. *Use these preliminary results to calculate* $\widehat{\alpha'''}_{FD}(\theta,h,n)$, S_{FD}^2 *and then estimate* $h_* = \left(\frac{9\sigma_{FD}^2}{4\alpha'''(\theta)^2}\right)^{1/5}$ *by* $\widehat{h}_* = \left(\frac{9S_{FD}^2}{4\widehat{\alpha'''}_{FD}(\theta,h,n)^2}\right)^{1/5}$
4. *Generate* $n \times \lceil T/\tau \rceil$ *values of* ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
5. *Use these random numbers to generate vectors containing* n *simulated prices of the ERBO, that is* $Y_i(\theta + k\hat{h}_*)$ *with* $k \in \{1, -1\}$

6. Use these results to calculate $\widehat{\alpha'}_{FD}(\theta, \hat{h}_*, n)$. ■

Next the PA for the FDA of the first derivative with respect to Π_0 (Delta) will be given; estimation of the sensitivity with respect to the other parameters follows in exactly the same way.

Algorithm 6.2 (PA for estimation of Delta using FDA)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{G}$ - *the sensitivity of the ERBPO with regard to* Π_0
SE - *the SE of the estimate*
CI - *95% CI*

Algorithm:

$h \leftarrow 0.1 \times \Pi_0$
$\underline{k} \leftarrow [3, 1, -1, -3]$
$P \leftarrow 0$ *[*$0.1n \times 4$ *matrix]*
$\underline{G}_1 \leftarrow \underline{0}$ *[vector of length* $0.1n$*]*
$\underline{G}_3 \leftarrow \underline{0}$ *[vector of length* $0.1n$*]*
for $m = 1$ *to* 4
  $\Pi_0^* \leftarrow \Pi_0$
  $\Pi_0 \leftarrow \Pi_0^* + \underline{k}[m] \times h$
  if $(T \bmod \tau = 0)$ *then*
    $\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
  else
    $\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \mod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
  end if
  for $l = 1$ *to* $0.1 \times n$
    $\Pi \leftarrow \Pi_0$
    for $j = 1$ *to* $\lceil T/\tau \rceil$
      generate ϵ_1 *and* ϵ_2 *using the IPIT and Equations 2.7 & 2.8*
      $\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i \sqrt{\underline{\Delta t}[j]}\right)$
    end for
    $P[l, m] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
  end for
end for
$\underline{G}_3[j] \leftarrow (P[j,1] - 3P[j,2] + 3P[j,3] - P[j,4])/(2h)^3$ *for* $j = 1, ..., 0.1n$
$\bar{G}_3 \leftarrow \sum_{i=1}^{0.1n} \underline{G}_3[i]/(0.1n)$
$\underline{G}_1[j] \leftarrow (P[j,2] - P[j,3])/(2h)$ *for* $j = 1, ..., 0.1n$
$\bar{G}_1 \leftarrow \sum_{i=1}^{0.1n} G_1[i]/(0.1n)$
$S_G \leftarrow \sqrt{\sum_{i=1}^{0.1n} (2h \times (G_1[i] - \bar{G}_1))^2/(0.1n - 1)}$

$h_* \leftarrow \left(\frac{9S_G^2}{4\bar{G}_3^2}\right)^{1/5}$
$P \leftarrow 0$ *[$n \times 2$ matrix]*
$\underline{G} \leftarrow \underline{0}$ *[vector of length n]*
$\underline{k} \leftarrow [1, -1]$
for $m = 1$ *to* 2
  $\Pi_0 \leftarrow \Pi_0^* + \underline{k}[m] \times h_*$
  if $(T \bmod \tau = 0)$ *then*
    $\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length T/τ)*
  else
    $\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \bmod \tau]$ *(length $\lceil T/\tau \rceil$)*
  end if
  for $l = 1$ *to* n
    $\Pi \leftarrow \Pi_0$
    for $j = 1$ *to* $\lceil T/\tau \rceil$
      generate ϵ_1 and ϵ_2 using the IPIT and Equations 2.7 & 2.8
      $\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i \sqrt{\underline{\Delta t}[j]}\right)$
    end for
    $P[l, m] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
  end for
end for
$\underline{G}[j] \leftarrow (P[j, 1] - P[j, 2])/(2h)$ *for* $j = 1, ..., n$
$\bar{G} \leftarrow \sum_{i=1}^{n} G[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$ ■

The most important loops are the m loops, where the value of the parameter under investigation is adjusted by $+\underline{k}[m] \times h$. This part of the algorithm needs to be adjusted when the parameter changes, that is

$$\begin{aligned} \Pi_0^* &\leftarrow \Pi_0 \\ \Pi_0 &\leftarrow \Pi_0^* + \underline{k}[m] \times h \end{aligned} \tag{6.5}$$

can be generalised to

$$\begin{aligned}\theta^* &\leftarrow \theta \\ \theta &\leftarrow \theta^* + \underline{k}[m] \times h \end{aligned} \tag{6.6}$$

where θ is the parameter for which sensitivity is being estimated.

For the second derivative with respect to θ, usually Π_0, no optimal h exists so that the sensitivity can simply be estimated by

$$\widehat{\alpha''}_{FD}(\theta, h, n) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i(\theta+h) - 2Y_i(\theta) + Y_i(\theta-h)}{h^2}\right) \tag{6.7}$$

for some small h, say $10\%\theta$.

6.1.2 Pathwise Derivative Estimates

The method of estimating the sensitivities with PDE is similar for all parameters, but the algorithm changes with each parameter (summarised in this section). As discussed in the literature review,

$$\widehat{\alpha'}_{PD}(\theta, n) \quad = \quad \frac{1}{n}\sum_{i=1}^{n} Y_i'(\theta),$$

is an unbiased estimator for $\alpha'(\theta)$, because $E[\widehat{\alpha'}_{PD}(\theta)] = \alpha'(\theta)$. The derivation of the derivative of $Y(\theta)$ with respect to $\theta \in \{\Pi_0, \sigma_i, r, \rho, T\}$ is given in Appendix C. The PDE method does not allow (due to the nature of the derivative) for estimating the gamma Greek (second order derivative with respect to Π_0) of an option (see Glasserman, 2004:392-393).

It is important to note that the algorithm used to calculate $Y_i'(\theta)$ will have to be adjusted in order to save time with the computation. This section will be split up into the different sensitivity parameters where each will be discussed and then the algorithm will be given. The following algorithm will be the main initial part for all the algorithms that follow. The output of this algorithm can be used in the algorithms that follow.

Algorithm 6.3 (ALGA: Main algorithm for PDE of ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\underline{P}$ *- vector containing simulated discounted payoffs [length n]*
$\underline{\Pi}_T$ *- vector containing simulated values of the rebalanced portfolio [length n]*
$\underline{\Delta t}$ *- vector containing difference between timestamps*

Z_1, Z_2 - $\lceil T/\tau \rceil \times n$ *random 2-dimensional* $MVNormal(\underline{0}, I)$ *variables with*
$[Z_1[i,j], Z_2[i,j]] \sim MVNormal(0,1)$ *[*$\lceil T/\tau \rceil \times n$ *matrices]*
ϵ_1, ϵ_2 - $\lceil T/\tau \rceil \times n$ *random correlated 2-dimensional* $MVNormal(\underline{0}, \Sigma)$ *variables*
$[\epsilon_1[i,j], \epsilon_2[i,j]] \sim MVNormal(0, \Sigma)$ *[*$\lceil T/\tau \rceil \times n$ *matrices]*
X_1, X_2 - *two matrices of size* $\lceil T/\tau \rceil \times n$ *used in other computations*
$\underline{\mathbb{1}}$ - *vector of size* n *used in other computations*

Algorithm:

$\underline{P} \leftarrow \underline{0}$ *[vector of length* n*] for* $i = 1, 2$
$Z_i, \epsilon_i, X_i \leftarrow 0$ *[matrix of size* $\lceil T/\tau \rceil \times n$*] for* $i = 1, 2$
if $(T \bmod \tau = 0)$ *then*
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
$\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \bmod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
generate ϵ_1 *and* ϵ_2 *using the IPIT (Definition 2.1) and*
Equations 2.7 & 2.8 with Z_1 *and* Z_2
for $l = 1$ *to* n
$\Pi \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
$X_i[j,l] \leftarrow v_i \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)\underline{\Delta t}[j] + \sigma_i \epsilon_i[j,l]\sqrt{\underline{\Delta t}[j]}\right)$ *for* $i = 1, 2$
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} X_i[j,l]$
end for
$\underline{\Pi}_T[l] \leftarrow \Pi$
$\underline{P}[l] \leftarrow \max\{K - \Pi, 0\} e^{-rT}$
$\underline{\mathbb{1}}[l] \leftarrow ifelse(K > \underline{\Pi}_T[l], -1, 0)$
end for ■

The output of the above algorithm will be used in the sections to follow. When using the function $ALGA(\cdot)$ inside another algorithm, the output can be used for further calculations.

6.1.2.1 Sensitivity with respect to Π_0

The derivative of $Y(\cdot)$ with respect to Π_0 is derived in Appendix C, given by Equation C.2 as

$$\frac{\mathrm{d}Y}{\mathrm{d}\Pi_0} = e^{-rT}(-\mathbb{1}_{\{K > \Pi_T(\Pi_0)\}})\frac{\Pi_T(\Pi_0)}{\Pi_0}. \tag{6.8}$$

The algorithm used to estimate the sensitivity with regards to Π_0 therefore is

Algorithm 6.4 (PA to estimate the sensitivity with respect to Π_0 with PDE of ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ

Output: $\bar{G}$ - *sensitivity of the price of the ERBPO with regard to* Π_0

SE - *the SE of the estimate*

CI - *95% CI*

Algorithm:

$\underline{G} \leftarrow \underline{0}$

$ALGA(n, \Pi_0, \tau, v_1, v_2, T, r, K, \sigma_1, \sigma_2, \rho)$

for $i = 1$ *to* n

$\underline{G}[i] \leftarrow e^{-rT} \times \underline{\mathbb{1}}[i] \times \frac{\underline{\Pi_T}[i]}{\Pi_0}$

end for

$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$

$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$

$SE \leftarrow S_G/\sqrt{n}$

$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$ ■

Figure 6.1 provides a graphical representation of Algorithm 6.4.

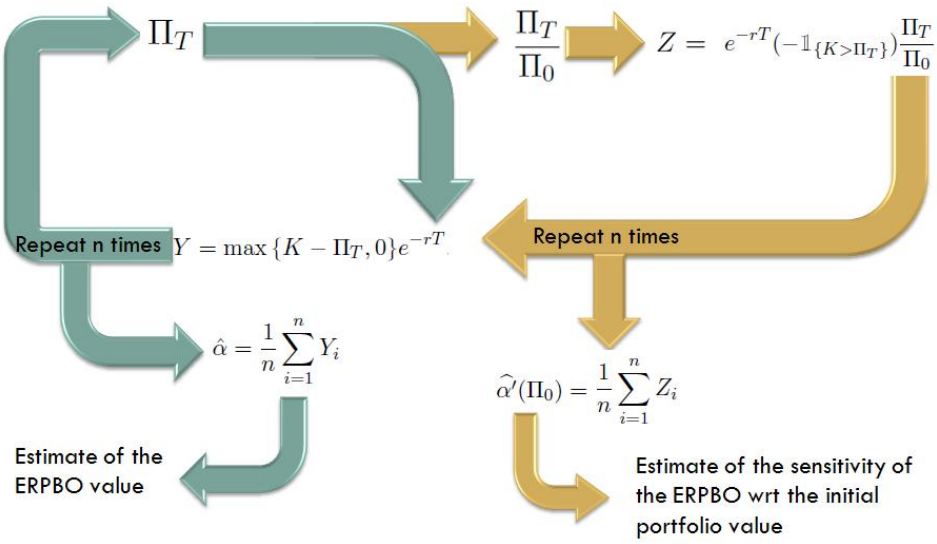

Figure 6.1: Graphical representation of Algorithm 6.4.

6.1.2.2 Sensitivity with respect to σ_i

The derivative of $Y(\cdot)$ with respect to σ_i is derived in Appendix C, given by Equation C.3 as

$$\begin{aligned}\frac{\mathrm{d}Y}{\mathrm{d}\sigma_i} &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\sigma_i)\}})\Pi_T(\sigma_i)\times \\ &\quad \left(\sum_{j=1}^{\lceil T/\tau\rceil}\frac{v_i\exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j+\sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)\left(-\sigma_i\Delta t_j+\epsilon_{i,j}\sqrt{\Delta t_j}\right)}{\sum_{l=1}^{2}v_l\exp\left(\left(r-\frac{\sigma_l^2}{2}\right)\Delta t_j+\sigma_l\epsilon_{l,j}\sqrt{\Delta t_j}\right)}\right) \\ &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\sigma_i)\}})\Pi_T(\sigma_i)\left(\sum_{j=1}^{\lceil T/\tau\rceil}\frac{X_{i,j}\left(-\sigma_i\Delta t_j+\epsilon_{i,j}\sqrt{\Delta t_j}\right)}{X_{1,j}+X_{2,j}}\right). \qquad (6.9)\end{aligned}$$

The algorithm used to estimate the sensitivity with regards to σ_* therefore is

Algorithm 6.5 (PA to estimate the sensitivity with respect to σ_* with PDE of ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{G}$ - *sensitivity of the price of the ERBPO with regard to* σ_*
SE - *the SE of the estimate*
CI - *95% CI*
Algorithm:
$\underline{G} \leftarrow \underline{0}$
ALGA(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*
for $i = 1$ *to* n
$Temp_3 \leftarrow 0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
$Temp_1 \leftarrow X_*[j,i]\left(-\sigma_*\underline{\Delta t}[j]+\epsilon_*[j,i]\sqrt{\underline{\Delta t}[j]}\right)$
$Temp_2 \leftarrow X_1[j,i]+X_2[j,i]$
$Temp_3 \leftarrow Temp_3 + Temp_1/Temp_2$
end for
$\underline{G}[i] \leftarrow e^{-rT}\times\underline{\mathbb{1}}[i]\times\underline{\Pi}_T[i]\times Temp_3$
end for
$\bar{G} \leftarrow \sum_{i=1}^{n}\underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n}(\underline{G}[i]-\bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G}-1.96\times SE, \bar{G}+1.96\times SE]$ ■

6.1.2.3 Sensitivity with respect to r

The derivative of $Y(\cdot)$ with respect to r is derived in Appendix C, given by Equation C.4 as

$$\frac{\mathrm{d}Y}{\mathrm{d}r} = Te^{-rT}\left(-[K-\Pi_T(r)]^+ + (-\mathbb{1}_{\{K>\Pi_T(r)\}})\Pi_T(r)\right). \tag{6.10}$$

The algorithm used to estimate the sensitivity with respect to r therefore is

Algorithm 6.6 (PA to estimate the sensitivity with respect to r with PDE of ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{G}$ - *sensitivity of the price of the ERBPO with regard to* r
SE - *the SE of the estimate*
CI - *95% CI*
Algorithm:
$\underline{G} \leftarrow \underline{0}$
$ALGA(n, \Pi_0, \tau, v_1, v_2, T, r, K, \sigma_1, \sigma_2, \rho)$
for $i = 1$ *to* n
$\underline{G}[i] \leftarrow -T \times \underline{P}[i] + e^{-rT}\underline{\mathbb{1}}[i] \times \underline{\Pi}_T[i] \times T$
end for
$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n}(\underline{G}[i]-\bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$ ■

6.1.2.4 Sensitivity with respect to ρ

The derivative of $Y(\cdot)$ with respect to ρ is derived in Appendix C, given by Equation C.5 as

$$\begin{aligned}\frac{\mathrm{d}Y}{\mathrm{d}\rho} &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\rho)\}})\Pi_T(\rho) \times \\ &\quad \left(\sum_{j=1}^{\lceil T/\tau \rceil} \frac{v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2 \epsilon_{2,j}\sqrt{\Delta t_j}\right)\left(\sigma_2\sqrt{\Delta t_j}\left(Z_{1,j} - \frac{\rho Z_{2,j}}{\sqrt{1-\rho^2}}\right)\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)}\right) \qquad (6.11)\\ &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\sigma_i)\}})\Pi_T(\sigma_i)\left(\sum_{j=1}^{\lceil T/\tau \rceil} \frac{X_{2,j}\left(\sigma_2\sqrt{\Delta t_j}\left(Z_{1,j} - \frac{\rho Z_{2,j}}{\sqrt{1-\rho^2}}\right)\right)}{X_{1,j}+X_{2,j}}\right). \qquad (6.12)\end{aligned}$$

The algorithm used to estimate the sensitivity with respect to ρ therefore is

Algorithm 6.7 (PA to estimate the sensitivity with respect to ρ with PDE of ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ

Output: $\bar{G}$ - *sensitivity of the price of the ERBPO with respect to* ρ

SE - *the SE of the estimate*

CI - *95% CI*

Algorithm:

$\underline{G} \leftarrow \underline{0}$

$ALGA(n, \Pi_0, \tau, v_1, v_2, T, r, K, \sigma_1, \sigma_2, \rho)$

for $i = 1$ *to* n

$Temp_3 \leftarrow 0$

for $j = 1$ *to* $\lceil T/\tau \rceil$

$Temp_1 \leftarrow X_2[j,i] \times \left(\sigma_2 \sqrt{\underline{\Delta t}[j]} \left(Z_1[j,i] - \frac{\rho Z_2[j,i]}{\sqrt{1-\rho^2}} \right) \right)$

$Temp_2 \leftarrow X_1[j,i] + X_2[j,i]$

$Temp_3 \leftarrow Temp_3 + Temp_1/Temp_2$

end for

$\underline{G}[i] \leftarrow e^{-rT} \times \underline{\mathbb{1}}[i] \times \underline{\Pi}_T[i] \times Temp_3$

end for

$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$

$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$

$SE \leftarrow S_G/\sqrt{n}$

$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$ ■

6.1.2.5 Sensitivity with respect to T

The derivative of $Y(\cdot)$ with respect to T is derived in Appendix C, given by Equation C.7 as

$$\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}T} &= -re^{-rT}[K - \Pi_T(T)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(T)\}})\Pi_T(T) \times \\
&\qquad \frac{\sum_{i=1}^{2} v_i \exp\left(\nu_i(T - \lfloor \frac{T}{\tau} \rfloor \tau) + \sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil} \sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}\right) \left(\nu_i + \frac{\sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil}}{2\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}}\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i(T - \lfloor \frac{T}{\tau} \rfloor \tau) + \sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil} \sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}\right)} \\
&= -re^{-rT}[K - \Pi_T(T)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(T)\}})\Pi_T(T) \times
\end{aligned}$$

$$\frac{\sum_{i=1}^{2} X_{i,\lceil \frac{T}{\tau} \rceil} \left(\nu_i + \frac{\sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil}}{2\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}} \right)}{\sum_{i=1}^{2} X_{i,\lceil \frac{T}{\tau} \rceil}} \tag{6.13}$$

for $T/\tau \notin \mathbb{Z}$, and cannot be derived with regards to T when $T/\tau \in \mathbb{Z}$.

The algorithm used to estimate the sensitivity with respect to T therefore is

Algorithm 6.8 (PA to estimate the sensitivity with respect to T with PDE of ERBPO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ

Output: $\bar{G}$ - *sensitivity of the price of the ERBPO with respect to* T

SE - *the SE of the estimate*

CI - *95% CI*

Algorithm:

$\underline{G} \leftarrow \underline{0}$

ALGA(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*

if $T \mod \tau \neq 0$ *then*

for $i = 1$ *to* n

$Temp_1 \leftarrow X_1[\lceil \frac{T}{\tau} \rceil, i] \times \left(\left(r - \frac{\sigma_1^2}{2} \right) + \frac{\sigma_1 \epsilon_1[\lceil \frac{T}{\tau} \rceil, i]}{2\sqrt{\underline{\Delta t}[\lceil \frac{T}{\tau} \rceil]}} \right)$

$Temp_2 \leftarrow X_2[\lceil \frac{T}{\tau} \rceil, i] \times \left(\left(r - \frac{\sigma_2^2}{2} \right) + \frac{\sigma_2 \epsilon_2[\lceil \frac{T}{\tau} \rceil, i]}{2\sqrt{\underline{\Delta t}[\lceil \frac{T}{\tau} \rceil]}} \right)$

$Temp_3 \leftarrow X_1[\lceil \frac{T}{\tau} \rceil, i] + X_2[\lceil \frac{T}{\tau} \rceil, i]$

$\underline{G}[i] \leftarrow -r \times \underline{P}[i] + e^{-rT} \mathbb{1}[i] \times \underline{\Pi}_T[i] \times (Temp_1 + Temp_2)/Temp_3$

end for

else

$\underline{G}[i] \leftarrow NA$

end if

$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$

$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$

$SE \leftarrow S_G/\sqrt{n}$

$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$ ■

6.2 RESULTS

The results of a simulation of the ERBPO sensitivities with respect to different parameters with the (a) FDA and (b) PDE methods are provided at the end of this chapter. The parameters that were used are $\Pi_0 = 1\ 000$, $K = 1\ 000$, $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $T \in \{5, 5.5\}$, $\rho = 0.5$, $\tau = 1$, $v_1 = v_2 = 0.5$ and with $n = 10^5$. The results will be discussed here, and concluded in the last section.

Tables 6.1 and 6.2 below compare the FDA and PDE methods when $T \bmod \tau = 0$. It is clear from the result tables that the point estimates and SEs for both methods are almost the same where both methods could be used. Those sensitivities that could be calculated with PDE had significantly shorter computing times. Note that CIs will be split up into the Lower Confident Bound (CB) and Upper CB, and that'$-$' indicates a really small computing time (close to 0).

Table 6.1: Point estimate, SE, CIs and Computing time (in seconds) of estimation of the sensitivities, using the FDA method, of an ERBPO with respect to different parameters. With $\Pi_0 = 1\ 000$, $K = 1\ 000$, $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $T = 5$, $\rho = 0.5$, $\tau = 1$, $v_1 = v_2 = 0.5$ and $n = 10^5$.

θ	Point Estimate	SE	Lower CB	Upper CB	Time (s)
Π_0	-0.2925	0.0010	-0.2944	-0.2906	2.47
Π_0^2	0.0006	0.0000	0.0005	0.0006	0.28
σ_1	334.8522	1.2267	332.4479	337.2564	2.49
σ_2	330.8911	1.1834	328.5717	333.2105	2.46
T	6.2389	0.3312	5.5897	6.8881	2.71
r	-2 200.3240	6.7797	-2 213.6130	-2 187.0360	2.50
ρ	64.8938	0.5066	63.9009	65.8867	2.51

Table 6.2: Point estimate, SE, CIs and Computing time (in seconds) of estimation of the sensitivities, using the PDE method, of an ERBPO with respect to different parameters. With $\Pi_0 = 1\ 000$, $K = 1\ 000$, $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $T = 5$, $\rho = 0.5$, $\tau = 1$, $v_1 = v_2 = 0.5$ and $n = 10^5$.

θ	Point Estimate	SE	Lower CB	Upper CB	Time (s)
Π_0	-0.2914	0.0010	-0.2934	-0.2895	-
Π_0^2	NA	NA	NA	NA	NA
σ_1	333.2542	1.2353	330.8330	335.6753	0.77
σ_2	335.2511	1.2403	332.8201	337.6821	0.75
T	NA	NA	NA	NA	NA
r	-2 209.3940	6.8021	-2 222.7260	-2 196.0620	-
ρ	64.4052	0.5099	63.4058	65.4046	0.84

Tables 6.3 and 6.4 on page 71 compare the FDA and PDE methods when $T \bmod \tau \neq 0$. The interpretation remains the same as for the previous results, except now the sensitivity of T can be calculated with the PDE method. For all of the results the PDE method, where it could be used, performed better in terms of computing time.

Table 6.3: Point estimate, SE, CIs and Computing time (in seconds) of estimation of the sensitivities, using the FDA method, of an ERBPO with respect to different parameters. With $\Pi_0 = 1\ 000$, $K = 1\ 000$, $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $T = 5.5$, $\rho = 0.5$, $\tau = 1$, $v_1 = v_2 = 0.5$ and $n = 10^5$.

θ	Point Estimate	SE	Lower CB	Upper CB	Time (s)
Π_0	-0.3115	0.0006	-0.3128	-0.3103	2.61
Π_0^2	0.0005	0.0000	0.0005	0.0005	0.30
σ_1	343.8951	1.2625	341.4207	346.3696	2.64
σ_2	343.8191	1.2699	341.3301	346.3080	2.67
T	6.1911	0.1581	5.8812	6.5009	2.78
r	-2 384.8950	7.3453	-2 399.2910	-2 370.4980	2.66
ρ	67.4878	0.4955	66.5167	68.4590	2.65

Table 6.4: Point estimate, SE, CIs and Computing time (in seconds) of estimation of the sensitivities, using the PDE method, of an ERBPO with respect to different parameters. With $\Pi_0 = 1\ 000$, $K = 1\ 000$, $\sigma_1 = \sigma_2 = 0.3$, $r = 0.03$, $T = 5.5$, $\rho = 0.5$, $\tau = 1$, $v_1 = v_2 = 0.5$ and $n = 10^5$.

θ	Point Estimate	SE	Lower CB	Upper CB	Time (s)
Π_0	-0.2820	0.0010	-0.2839	-0.2802	-
Π_0^2	NA	NA	NA	NA	NA
σ_1	344.1618	1.2703	341.6720	346.6517	0.78
σ_2	344.8239	1.2704	342.3339	347.3139	0.74
T	5.4983	0.2296	5.0483	5.9483	0.04
r	-2 392.5650	7.3711	-2 407.0130	-2 378.1180	0.02
ρ	66.8931	0.5180	65.8777	67.9085	0.86

6.3 CONCLUSION

This chapter provides different methods for estimating sensitivities of the ERBPO with respect to different parameters. The results presented in the previous section indicate that the PDE method is superior. The PDE method, however, cannot be used when estimating the second order sensitivity with respect to Π_0, and the sensitivity with respect to T when $T/\tau \in \mathbb{Z}$. The FDA method should be used in these instances.

CHAPTER 7

CONCLUSION AND OPEN QUESTIONS

MC techniques can be used as a method to price a variety of different exotic options. This research aimed to find the best MC technique to price the ERBO. The ERBO, an option on a portfolio that gets rebalanced on a fixed basis, is a new type of option on which, to the author's best knowledge, no current literature exists.

General MC was used to price the ERBPO by using a simplistic as well as a refined (obtained with the help of a mathematical proof) method. The refined method delivered much smaller computational time compared to the simplistic method, and was therefore chosen as the MC method to be made more efficient in the research that followed.

Different VRTs were applied to the refined MC method and compared in terms of a fixed computational time and SE. The results showed that initially AVs outperformed CVs, but not by a significant amount. Next, the most efficient method was chosen only when it outperformed the second best method by a certain amount. Here CVs outperformed all the other methods. Therefore CVs, with the sum of two vanilla put/call options as the CV, was chosen as the most efficient method.

QMC techniques can be used to increase the convergence of the estimator, and in some cases reduce the SE as well. Different LDSs were applied to the refined MC method and results were compared. It was found that the best LDS to use in simulation is the SS with Owen type as well as Faure-Tezuka type scrambling. This can be combined with the VRT simply by using this LDS as the random number generator for the $\underline{U} \sim Unif(0,1)^d$ random variables used in the simulation.

In Chapter 6 estimation of sensitivities of the ERBPO with regard to different input parameters was discussed. It used both FDA and PDE for estimation. The results showed that when estimating sensitivities the PDE approach outperforms the FDA method. However, the FDA method should be used where PDE cannot.

In Appendix D the final combined algorithm for optimal pricing of the ERBO and estimation of its sensitivities is given. This can be programmed in any mathematical/statistical package. This algorithm was programmed in `R` (R Development Core Team, 2010) and the code can be found in Appendix E.[1]

Although the initial research question was answered, there still exist some open questions which can serve as future research topics, the most important being whether there exists a closed-form solution for this problem. The ERBO can possibly also be extended from two assets to m assets, where $m > 2$. Further research could be done on the combination of different VRTs to possibly find an improvement on the classical VRTs. The most interesting open question is whether a method on how to predict which VRT would reduce the SE the most can be found by considering the input parameters first. This could be done with Linear Discrimenant Analysis, Classification and Regression Trees or other Data-mining techniques on previous simulations. An example for this could be different pricing techniques for in-the-money, at-the-money, and out-of-the-money options.

Finally, this research shows that the refined MC method decreased the computational time of the value of the ERBO by ±98%; the error of the estimates was smaller than it was for normal MC ±95% of the time using different VRTs; and by applying QMC methods the amount of simulations needed to obtain the same accuracy of normal MC decrease by ±99%. Furthermore, 75% of the sensitivity estimators was improved in terms of bias and their computing time also decreased by ±95% compared to the old method. Therefore, using advanced simulation procedures is worthwhile to implement when pricing exotic type derivatives using MC simulation.

[1] Only the pricing of the ERBPO was discussed in this research, the algorithm was adapted to price the ERBCO as well in Appendix D.

APPENDIX A

DEFINITION AND VALUATION FRAMEWORK FOR OPTIONS

A brief revision of the idea behind options and the framework in which they are valued, including a discussion of the definition of an option and the calculation aspects thereof will be provided in this appendix. The most important options are stock options where stocks form the underlying of the option. The properties of these stock options will be described in the following subsection, including the valuation framework and assumptions which are made when valuating them. The last subsection provides the derivation for the process, or path followed by stocks. This is key to the MC numerical method for valuing options.

A.1 DEFINITION OF AN OPTION

In finance, an option is a contract between a buyer and a seller. The contract gives the buyer the 'option' or the right, but not the obligation, to buy or to sell a specified underlying on, or before, the option's expiration time, at an agreed price - the strike price. The seller of the option will in turn require a payment from the buyer for granting this opportunity, which is called the premium or price. Selling an option is referred to as 'writing' an option or taking a 'short' position. When purchasing an option one enters into a 'long' position. A few distinctions will be made between the main types of options. The first is the difference between when the option is exercised: on (European type) or before (American type) the expiration date. Secondly, the individual with the long position can either buy (call option) or sell (put option) the underlying from or to the individual with the short position. Different underlyings exists on which an option can be written. This subsection is based on the work adapted from Hull (2009:179-184) and Wilmott (2007:28-44).

The first distinction of options is made between European and American type options. European options can be exercised only on the expiration date itself, whereas American options can be exercised at any time prior to the expiration date. The ERBO is an option that can only be exercised on the

expiration date, therefore the particular interest of this study is not applicable to American type options and it will not be discussed any further in this research. All options that will be discussed further are European type options.[1]

A call option gives the individual with the long position the right to buy the underlying at the strike price, whereas a put option gives him the right to sell the underlying at the strike price to the writer of the option. Depending on whether it is a call or put option, if the individual with the long position chooses to exercise this right, the writer is obliged to sell or buy the underlying at the agreed price. The individual with the long position may also choose not to exercise this right, and no exchange will take place.

When the payoff of a European call or put option ($\max\{\Psi_T - K, 0\}$ or $\max\{K - \Psi_T, 0\}$) only depends on the closing price of a stock at time T, it is also known as a plain vanilla option - that is $\Psi_T = S_T$. Different variations for these underlyings exist and when this changes, the option moves from being a plain vanilla to become an exotic option. The underlying of an option can be anything from a variety of assets - the most common being a security (stock or bond). In some cases the underlying can be a combination of more than one asset - multi-asset options - or in other cases the value of the underlying at time T, Ψ_T, can depend on the path followed by this asset. The best example to describe the latter is to give the definition of an Asian option which is a type of exotic option: for Asian options the payoff is determined by the average price followed by the underlying's price over the life of the option. This is different from of the plain vanilla option, where the payoff of the option depends only on the price of the underlying at maturity. This is called a path-dependent option. Path-dependent and multi-asset options are examples of exotic options as their underlyings are somewhat different from the plain vanilla option.

Consider a plain vanilla option. Let $\Psi_T = S_T$ denote the closing price of a stock at time T, and K denote the strike price for the option. If one considers a long call, then the payoff of the holder at time T will depend on S_T and K. If $S_T > K$ the holder will buy the underlying stock at price K from the writer and sell it immediately in the market for S_T making a profit of $S_T - K$. If $S_T < K$ the holder will let the option expire worthless. Therefore the payoff at time T for the individual with a long position in a plain vanilla call can be expressed as $\max\{S_T - K, 0\}$. Similarly the payoff for the long position in a put option can be expressed as $\max\{K - S_T, 0\}$. The payoff for an individual holding a short position in any of these two options will be the negative of these expressions.

The most important part of options is the premium that needs to be charged for this contract. Although closed-form solutions exist to price the premium for plain vanilla options, pricing other exotic type options becomes more complicated. In option pricing theory there exist several techniques to determine the correct premium to charge for an option, one of which is MC - literature on MC numerical methods will be discussed in depth in Chapter 2. Next, the assumptions that are made in the option pricing framework for stocks need to be discussed, after which processes for stocks will be considered.

[1] Bermuden type options can be exercised on or before expiry, but only on predetermined dates.

A.2 PROPERTIES AND ASSUMPTIONS OF STOCK OPTIONS

In the option pricing framework when underlyings are stocks, there are some assumptions that need to be made to price these options.[2] The assumptions that are made for options on stocks are as follows:

- There are no transaction costs.
- All trading profits (net of trading losses) are subject to the same tax rate.
- Borrowing and lending are both possible at the risk-free interest rate.
- No arbitrage - market participants are prepared to take advantage of arbitrage opportunities as they arise.

The option pricing framework uses Risk-neutral valuation. In a risk neutral world all individuals are indifferent to risk. In such a world investors do not need compensation for risk, and the expected return on all investments are just the risk-free rate. Risk-neutral valuation assumes that the world is risk-neutral when pricing options. The resulting prices are correct not just in a risk-neutral world, but in other worlds as well. Therefore, when working with any stock and calculating the price of the option, the expected return can be ignored, and just be substituted with the risk-free rate.

A.3 WIENER PROCESSES AND ITÔ'S LEMMA

This section derives the stochastic process that is followed by stock prices and which forms the basis of all pricing theory for stock options. The theory in this subsection is adapted from Hull (2009:259-270), Wilmott (2007:117-136) and Chan and Wong (2006:11-28).

Any variable whose value changes randomly over time with uncertainty, is said to follow a stochastic process. Stochastic processes can be classified as discrete- or continuous-time processes. A discrete-time stochastic process is one where the value of the variable can change only at certain fixed points in time, whereas a continuous-time stochastic process is one where changes can take place at any time. Stochastic processes can also be classified as continuous- or discrete-variable processes. In a continuous-variable process, the underlying variable can take on any value within a certain range, whereas in a discrete variable process, only certain discrete values are possible. Note that in practice stock prices are not observed to follow continuous-variable, continuous-time processes, but rather discrete for both. However, in option pricing theory it is assumed that stock prices do follow the former.

Stock prices are said to have the Markov property which means that all previous information is contained in the current price. Predictions of future prices are uncertain and must be expressed in

[2]For more on the properties and assumptions of stock options, see Hull (2009:201-216) and Wilmott (2007:39-57).

terms of probability distributions. The Markov property implies that the probability distribution of the price at any particular future time is not dependent on the particular path followed by the price in the past. This property of stock prices is consistent with the weak form of market efficiency. This states that the present price of a stock impounds all the information contained in a record of past prices.

Consider now a stock that follows a Markov stochastic process. Suppose that its current value is X, and that the change in its value during 1 year is distributed $N(\mu, \sigma^2)$. The change in the value of the variable during any time period of length T is distributed $N(\mu T, \sigma^2 T)$, and specifically during a very short time period of length Δt is distributed $N(\mu \Delta t, \sigma^2 \Delta t)$. This follows from the additive nature of independent normally distributed variables.

The next concept to introduce is that of Wiener processes. The process followed by the variable described above is known as a Wiener process. It is a particular type of Markov stochastic process with a mean change of zero and variance rate of 1.0 per year. Expressed formally, a variable Z follows a Wiener process if it has the following two properties:

1. The change in Z during a small period of time Δt is $\Delta Z = \epsilon\sqrt{\Delta t}$ where ϵ has a standardised normal distribution $N(0, 1)$.
2. The values of ΔZ for any two non-overlapping short intervals of time, Δt, are independent.

Consider the following model

$$W(t_{k+1}) = W(t_k) + \epsilon_{t_k}\sqrt{\Delta t},$$

with $t_{k+1} - t_k = \Delta t$, and $k = 0, ..., N$ with $t_0 = 0$. The ϵ_{t_k}s are identically independently distributed $N(0, 1)$ random variables. From this equation the difference between two values at different time steps j and k with $j < k$ can be calculated as

$$W(t_k) - W(t_j) = \sum_{i=j}^{k-1} \epsilon_{t_i}\sqrt{\Delta t}.$$

The above equation implies the following three important results:

1. Because the right-hand side is a sum of normal independent random variables, it means that $\Delta W = W(t_k) - W(t_j)$ is also normally distributed.
2. By taking expectations:

$$E[W(t_k) - W(t_j)] \quad = \quad 0$$

$$\begin{aligned} Var(W(t_k) - W(t_j)) &= Var\left(\sum_{i=j}^{k-1} \epsilon_{t_i}\sqrt{\Delta t}\right) \\ &= E\left[\left(\sum_{i=j}^{k-1} \epsilon_{t_i}\sqrt{\Delta t}\right)^2\right] \\ &= (k-j)\Delta t \\ &= t_k - t_j. \end{aligned}$$

Hence $W(t_k) - W(t_j)$ is distributed $N(0, t_k - t_j)$.

3. For $t_i < t_j \leq t_k < t_l$, $W(t_l) - W(t_k)$ is uncorrelated with $W(t_j) - W(t_i)$ for any $i < j \leq k < l$.

Furthermore, if $\Delta t \to 0$ then ΔW can be written as $\mathrm{d}W$.

This can be generalised to form the generalised Wiener process which is described as $\Delta x = a\Delta t + b\Delta z = a\Delta + b\epsilon\sqrt{\Delta t}$, or more formally: the change in x during a small time interval Δt is equal to the drift of x, a, multiplied by Δt plus a random shock $b\Delta z \sim N(0, b^2\Delta t)$. When $\Delta t \to 0$, this becomes

$$\mathrm{d}x = a\,\mathrm{d}t + b\,\mathrm{d}z.$$

The process followed by a stock can now be defined with the help of the generalised Wiener process. The first obstacle to consider is the fact that a stock does not grow with a fixed value, it grows with a percentage. Therefore $\Delta S = \mu S\Delta t$, with μ the expected rate of return for a stock with price S. This becomes $\mathrm{d}S = \mu S\,\mathrm{d}t$ in the limit where $\Delta t \to 0$. Unfortunately in practice a stock does not move exactly like this, a shock with each jump needs to be added. Therefore the process for a stock becomes: $\mathrm{d}S = \mu S\,\mathrm{d}t + \sigma S\,\mathrm{d}z$.

When the process for S with jumps of size Δt is considered, the above equation becomes:

$$\begin{aligned} & \Delta S = \mu S\Delta t + \sigma S\Delta z \\ \Rightarrow \quad & \tfrac{\Delta S}{S} = \mu\Delta t + \sigma\Delta z \\ \Rightarrow \quad & \tfrac{\Delta S}{S} = \mu\Delta t + \sigma\epsilon\sqrt{\Delta t} \\ \Rightarrow \quad & \tfrac{\Delta S}{S} \sim N(\mu\Delta t; \sigma^2\Delta t) \end{aligned}$$

Although the process for the price of a stock (S) and the distribution of normal returns $\frac{\Delta S}{S}$ are now known, there still exists a problem. It will be shown why it is necessary to derive the process for $\ln S$. It is known that $\frac{\Delta S}{S} \sim N(\mu\Delta t; \sigma^2\Delta t)$, but $\frac{S_T - S_0}{S_0}$, the return over period T, is only approximately normally distributed with mean μT and standard deviation $\sigma\sqrt{T}$. This is due to the fact that $\frac{S_T - S_0}{S_0} \neq \sum \frac{\Delta S}{S}$.

Hence, let $\Delta \ln S = \ln S_{t+\Delta t} - \ln S_t$. An exact distribution can be found for $\ln S_T - \ln S_0$ because $\ln \frac{S_T}{S_0} = \sum \Delta \ln S$. Itô's lemma is used to derive the distribution for $\ln S$. Itô's lemma states that if the process $\mathrm{d}x = a(x,t)\,\mathrm{d}t + b(x,t)\,\mathrm{d}z$ is known, then the process of $G(x,t)$ can be expressed as follows:

$$\mathrm{d}G = \left(\frac{\partial G}{\partial x}a + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial x^2}b^2\right)\mathrm{d}t + \frac{\partial G}{\partial x}b\,\mathrm{d}z.$$

Therefore if $x \equiv S$ and $G(S,t) = \ln S$, the process for $\ln S$ can be expressed as $\mathrm{d}\ln S = \left(\mu - \frac{\sigma^2}{2}\right)\mathrm{d}t + \sigma\,\mathrm{d}z$. Then $\Delta \ln S \sim N\left(\left(\mu - \frac{\sigma^2}{2}\right)\Delta T; \sigma^2 \Delta T\right)$. This implies that

$$\begin{aligned} \Rightarrow \quad & \ln\left(\tfrac{S_T}{S_0}\right) = \ln S_T - \ln S_0 \sim N\left(\left(\mu - \tfrac{\sigma^2}{2}\right)T; \sigma^2 T\right) \\ \Rightarrow \quad & \ln S_T \sim N\left(\ln S_0 + \left(\mu - \tfrac{\sigma^2}{2}\right)T; \sigma^2 T\right). \end{aligned} \tag{A.1}$$

The final result is an exact distribution for the natural logarithms of the stock price at time T that is only dependent on the initial price S_0, μ, σ, and T. This result is better than the approximation that was derived earlier. The distribution of $\ln S_T$ forms the foundation for the BS Model. Only the final result will be given in the next section.

A.4 BLACK-SCHOLES OPTION PRICING FORMULAE

The BS Model (Black and Scholes, 1973) is one of the most important concepts in modern financial theory. It was developed in 1973 by Fisher Black, Robert Merton and Myron Scholes and is still widely used today, and regarded as one of the best ways of determining fair prices of vanilla options.

The value of a call (C) and put (P) option for a non-dividend paying underlying stock (S) with volatility σ and with strike price K, time till maturity T and risk-free rate r in terms of the BS parameters are:

$$BS_c = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2) \tag{A.2}$$

$$BS_p = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1) \tag{A.3}$$

with

$$\begin{aligned} d_1 &= \frac{\ln(\frac{S_0}{K}) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}} \\ d_2 &= d_1 - \sigma\sqrt{T}, \end{aligned}$$

and Φ the cumulative standard normal distribution function.

APPENDIX B

PROOF FOR REFINED MONTE CARLO FORMULA

B.1 THEOREM

Given a risk-free rate r and a portfolio Π with underlying assets S_i that is rebalanced every τ years to fixed proportions v_i for $i = 1, 2$, where each asset has a volatility σ_i and the correlation between these two assets are ρ. Then the value of the portfolio just before the j^{th} rebalance takes place, can be simulated with

$$\Pi^*_{j\tau} = \Pi_0 \prod_{l=1}^{j} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t + \sigma_i \epsilon_i \sqrt{\Delta t} \right) \right] \tag{B.1}$$

with

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix} \sim MVN_2 \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}; \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right). \tag{B.2}$$

B.2 PROOF

It will be shown that

$$\Pi^*_{j\tau} = \Pi_0 \prod_{l=1}^{j} \left[v_1 \frac{S_{1,l\tau}}{S_{1,(l-1)\tau}} + v_2 \frac{S_{2,l\tau}}{S_{2,(l-1)\tau}} \right] \tag{B.3}$$

holds for all $j = 1, 2, ..., \lfloor T/\tau \rfloor$.

This will be done with the help of mathematical induction. Therefore, it will be shown that the above equation is true for $j = 1$; assume that it is true for $j = k$; then prove that it holds for $j = k + 1$.

It is known that

$$\Pi_0 = w_{1,0} S_{1,0} + w_{2,0} S_{2,0} \tag{B.4}$$

with

$$w_{i,0} = \frac{v_i \Pi_0}{S_{i,0}}, \quad i = 1, 2. \tag{B.5}$$

Therefore,

$$\begin{aligned} \Pi_0 &= \frac{v_1 \Pi_0}{S_{1,0}} S_{1,0} + \frac{v_2 \Pi_0}{S_{2,0}} S_{2,0} \\ &= v_1 \Pi_0 + v_2 \Pi_0 \\ &= v_1 \Pi_0 + (1 - v_1) \Pi_0 \\ &= \Pi_0. \end{aligned} \tag{B.6}$$

The value of the portfolio just prior to the first rebalance is

$$\begin{aligned} \Pi^*_{1\tau} &= w_{1,0} S_{1,1\tau} + w_{2,0} S_{2,1\tau} \\ &= \frac{v_1 \Pi_0}{S_{1,0}} S_{1,1\tau} + \frac{v_2 \Pi_0}{S_{2,0}} S_{2,1\tau} \\ &= v_1 \Pi_0 \frac{S_{1,1\tau}}{S_{1,0}} + v_2 \Pi_0 \frac{S_{2,1\tau}}{S_{2,0}} \end{aligned}$$

$$= \Pi_0 \left[v_1 \frac{S_{1,1\tau}}{S_{1,0}} + v_2 \frac{S_{2,1\tau}}{S_{2,0}} \right]. \tag{B.7}$$

Now, asume that Equation B.3 holds for $j = k$, i.e.

$$\Pi^*_{k\tau} = \Pi_0 \prod_{l=1}^{k} \left[v_1 \frac{S_{1,l\tau}}{S_{1,(l-1)\tau}} + v_2 \frac{S_{2,l\tau}}{S_{2,(l-1)\tau}} \right] \tag{B.8}$$

and prove that it holds for $j = k + 1$.

For $j = k + 1$

$$\begin{aligned}
\Pi^*_{(k+1)\tau} &= w_{1,k\tau} S_{1,(k+1)\tau} + w_{2,k\tau} S_{2,(k+1)\tau} \\
&= \frac{v_1 \Pi^*_{k\tau}}{S_{1,k\tau}} S_{1,(k+1)\tau} + \frac{v_2 \Pi^*_{k\tau}}{S_{2,k\tau}} S_{2,(k+1)\tau} \\
&= \Pi^*_{k\tau} \left[v_1 \frac{S_{1,(k+1)\tau}}{S_{1,k\tau}} + v_2 \frac{S_{2,(k+1)\tau}}{S_{2,k\tau}} \right] \\
&= \Pi_0 \prod_{l=1}^{k} \left[v_1 \frac{S_{1,l\tau}}{S_{1,(l-1)\tau}} + v_2 \frac{S_{2,l\tau}}{S_{2,(l-1)\tau}} \right] \left[v_1 \frac{S_{1,(k+1)\tau}}{S_{1,k\tau}} + v_2 \frac{S_{2,(k+1)\tau}}{S_{2,k\tau}} \right] \quad \text{from Equation B.8} \\
&= \Pi_0 \prod_{l=1}^{k+1} \left[v_1 \frac{S_{1,l\tau}}{S_{1,(l-1)\tau}} + v_2 \frac{S_{2,l\tau}}{S_{2,(l-1)\tau}} \right].
\end{aligned} \tag{B.9}$$

Therefore,

$$\Pi^*_{j\tau} = \Pi_0 \prod_{l=1}^{j} \left[v_1 \frac{S_{1,l\tau}}{S_{1,(l-1)\tau}} + v_2 \frac{S_{2,l\tau}}{S_{2,(l-1)\tau}} \right] \tag{B.10}$$

holds for all $j = 1, 2, ..., \lfloor T/\tau \rfloor$. The above formula can now be refined by substituting in the formulae used to simulate $S_{i,j\tau}$. First, consider the fraction $\frac{S_{i,l\tau}}{S_{i,(l-1)\tau}}$ with $i = 1, 2$ and l any timestamp on the timeline, with the difference in timestamps l and $l - 1$ as Δt. Now, for any t and Δt,

$$\begin{aligned}
\frac{S_{i,l\tau}}{S_{i,(l-1)\tau}} &= \frac{S_{i,t+\Delta t}}{S_{i,t}} \\
&= \frac{S_{i,t} \exp\left(\left(r - \sigma_i^2/2 \right) \Delta t + \sigma_i \epsilon_i \sqrt{\Delta t} \right)}{S_{i,t}} \\
&= \exp\left(\left(r - \sigma_i^2/2 \right) \Delta t + \sigma_i \epsilon_i \sqrt{\Delta t} \right).
\end{aligned} \tag{B.11}$$

Therefore,

$$\Pi^*_{j\tau} = \Pi_0 \prod_{l=1}^{j} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t + \sigma_i \epsilon_i \sqrt{\Delta t} \right) \right] \tag{B.12}$$

with

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix} \sim MVN_2 \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}; \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right). \tag{B.13}$$

APPENDIX C

CHAPTER 6 DERIVATIVES

The value of the ERBO put is equal to $E[Y] = \alpha$, with

$$\begin{aligned} Y &= e^{-rT} \max\{K - \Pi_T, 0\} \\ &= e^{-rT}[K - \Pi_T]^+, \end{aligned}$$

with Π_T the value of the rebalanced portfolio at time T. A refined formula for Π_T was found in Chapter 3,

$$\begin{aligned} \Pi_T(\cdot) &= \Pi_0 \prod_{j=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t_j + \sigma_i \epsilon_{i,j} \sqrt{\Delta t_j} \right) \right] \\ &= \Pi_0 \prod_{j=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j} \sqrt{\Delta t_j} \right) \right] \end{aligned}$$

with $\Delta t = [\tau, \tau, ..., \tau, T \bmod \tau] = [\tau, \tau, ..., \tau, T - \lfloor \frac{T}{\tau} \rfloor \tau]$. This appendix gives the derivation of the first derivatives of the function Y with respect to Π_0, σ_1, T, r and ρ; as well as the second derivative with respect to Π_0. All derivations for a put option - the derivations for a call option follow in the same way.

Also note that

$$\frac{\mathrm{d}}{\mathrm{d}x} \prod_{j=1}^{k} f_j(x) = \left(\prod_{j=1}^{k} f_j(x) \right) \left(\sum_{j=1}^{k} \frac{f_j'(x)}{f_j(x)} \right) \tag{C.1}$$

with $f'(x)$ the derivative of the function $f(x)$ with respect to x.

The main result that will be used throughout is

$$\begin{aligned}\frac{\mathrm{d}[K-\Pi_T(x)]^+}{\mathrm{d}x} &= \frac{\mathrm{d}[K-\Pi_T(x)]^+}{\mathrm{d}\Pi_T(x)}\frac{\mathrm{d}\Pi_T(x)}{\mathrm{d}x}\\ &= -\mathbb{1}_{\{K>\Pi_T(x)\}}\frac{\mathrm{d}\Pi_T(x)}{\mathrm{d}x}.\end{aligned}$$

C.1 $\frac{\mathrm{d}Y}{\mathrm{d}\Pi_0}$

$$\frac{\mathrm{d}Y}{\mathrm{d}\Pi_0} = e^{-rT}\frac{\mathrm{d}[K-\Pi_T(\Pi_0)]^+}{\mathrm{d}\Pi_T(\Pi_0)}\frac{\mathrm{d}\Pi_T(\Pi_0)}{\mathrm{d}\Pi_0}.$$

The second part of this derivative, $\frac{\mathrm{d}\Pi_T(\Pi_0)}{\mathrm{d}\Pi_0}$, is

$$\begin{aligned}\frac{\mathrm{d}\Pi_T(\Pi_0)}{\mathrm{d}\Pi_0} &= \frac{\mathrm{d}}{\mathrm{d}\Pi_0}\Pi_T(\Pi_0)\\ &= \frac{\mathrm{d}}{\mathrm{d}\Pi_0}\Pi_0\prod_{j=1}^{\lceil T/\tau\rceil}\left[\sum_{i=1}^{2}v_i\exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j+\sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)\right]\\ &= \prod_{j=1}^{\lceil T/\tau\rceil}\left[\sum_{i=1}^{2}v_i\exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j+\sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)\right],\end{aligned}$$

so that

$$\begin{aligned}\frac{\mathrm{d}Y}{\mathrm{d}\Pi_0} &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\Pi_0)\}})\prod_{j=1}^{\lceil T/\tau\rceil}\left[\sum_{i=1}^{2}v_i\exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j+\sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)\right]\\ &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\Pi_0)\}})\frac{\Pi_T(\Pi_0)}{\Pi_0}.\end{aligned} \tag{C.2}$$

C.2 $\frac{\mathrm{d}Y}{\mathrm{d}\sigma_i}$

$$\begin{aligned}\frac{\mathrm{d}Y}{\mathrm{d}\sigma_i} &= e^{-rT}\frac{\mathrm{d}[K-\Pi_T(\sigma_i)]^+}{\mathrm{d}\Pi_T(\sigma_i)}\frac{\mathrm{d}\Pi_T(\sigma_i)}{\mathrm{d}\sigma_i}\\ &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\sigma_i)\}})\frac{\mathrm{d}\Pi_T(\sigma_i)}{\mathrm{d}\sigma_i}.\end{aligned}$$

The second part of this derivative, $\frac{\mathrm{d}\Pi_T(\sigma_i)}{\mathrm{d}\sigma_i}$, is

$$\begin{aligned}
\frac{\mathrm{d}\Pi_T(\sigma_i)}{\mathrm{d}\sigma_i} &= \frac{\mathrm{d}}{\mathrm{d}\sigma_i}\Pi_T(\sigma_i) \\
&= \frac{\mathrm{d}}{\mathrm{d}\sigma_i}\Pi_0 \prod_{j=1}^{\lceil T/\tau \rceil}\left[\sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\right] \\
&= \Pi_0 \frac{\mathrm{d}}{\mathrm{d}\sigma_i}\prod_{j=1}^{\lceil T/\tau \rceil} f_j(\sigma_i).
\end{aligned}$$

Next, the derivative of $f_j(\sigma_i)$ needs to be calculated in order to apply Equation C.1, i.e.

$$\begin{aligned}
f_j(\sigma_i) &= \sum_{l=1}^{2} v_l \exp\left(\left(r-\frac{\sigma_l^2}{2}\right)\Delta t_j + \sigma_l \epsilon_{l,j}\sqrt{\Delta t_j}\right) \\
\Rightarrow f_j'(\sigma_i) &= v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\left(-\sigma_i \Delta t_j + \epsilon_{i,j}\sqrt{\Delta t_j}\right),
\end{aligned}$$

so that

$$\begin{aligned}
\frac{\mathrm{d}\Pi_T(\sigma_i)}{\mathrm{d}\sigma_i} &= \Pi_0\left(\prod_{j=1}^{\lceil T/\tau \rceil}\left[\sum_{l=1}^{2} v_l \exp\left(\left(r-\frac{\sigma_l^2}{2}\right)\Delta t_j + \sigma_l \epsilon_{l,j}\sqrt{\Delta t_j}\right)\right]\right)\left(\sum_{j=1}^{\lceil T/\tau \rceil}\frac{f_j'(\sigma_i)}{f_j(\sigma_i)}\right) \\
&= \Pi_T(\sigma_i)\left(\sum_{j=1}^{\lceil T/\tau \rceil}\frac{v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\left(-\sigma_i \Delta t_j + \epsilon_{i,j}\sqrt{\Delta t_j}\right)}{\sum_{l=1}^{2} v_l \exp\left(\left(r-\frac{\sigma_l^2}{2}\right)\Delta t_j + \sigma_l \epsilon_{l,j}\sqrt{\Delta t_j}\right)}\right).
\end{aligned}$$

Therefore, for $i = 1, 2$

$$\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}\sigma_i} &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\sigma_i)\}})\Pi_T(\sigma_i) \times \\
&\quad \left(\sum_{j=1}^{\lceil T/\tau \rceil}\frac{v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\left(-\sigma_i \Delta t_j + \epsilon_{i,j}\sqrt{\Delta t_j}\right)}{\sum_{l=1}^{2} v_l \exp\left(\left(r-\frac{\sigma_l^2}{2}\right)\Delta t_j + \sigma_l \epsilon_{l,j}\sqrt{\Delta t_j}\right)}\right). \qquad \text{(C.3)}
\end{aligned}$$

C.3 $\frac{\mathrm{d}Y}{\mathrm{d}r}$

$$\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}r} &= [K-\Pi_T(r)]^+\frac{\mathrm{d}}{\mathrm{d}r}(e^{-rT}) + e^{-rT}\frac{\mathrm{d}}{\mathrm{d}r}[K-\Pi_T(r)]^+ \\
&= -Te^{-rT}[K-\Pi_T(r)]^+ + e^{-rT}\frac{\mathrm{d}}{\mathrm{d}r}[K-\Pi_T(r)] \\
&= -Te^{-rT}[K-\Pi_T(r)]^+ + e^{-rT}\frac{\mathrm{d}[K-\Pi_T(x)]^+}{\mathrm{d}\Pi_T(r)}\frac{\mathrm{d}\Pi_T(r)}{\mathrm{d}r} \\
&= -Te^{-rT}[K-\Pi_T(r)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(r)\}})\frac{\mathrm{d}\Pi_T(r)}{\mathrm{d}r}.
\end{aligned}$$

The second part of this derivative, $\frac{\mathrm{d}\Pi_T(r)}{\mathrm{d}r}$, is

$$\begin{aligned}
\frac{\mathrm{d}\Pi_T(r)}{\mathrm{d}r} &= \frac{\mathrm{d}}{\mathrm{d}r}\Pi_T(r) \\
&= \frac{\mathrm{d}}{\mathrm{d}r}\Pi_0\prod_{j=1}^{\lceil T/\tau\rceil}\left[\sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)\right] \\
&= \Pi_0\frac{\mathrm{d}}{\mathrm{d}r}\prod_{j=1}^{\lceil T/\tau\rceil} f_j(r).
\end{aligned}$$

Next, the derivative of $f_j(r)$ needs to be calculated in order to apply Equation C.1, i.e.

$$\begin{aligned}
f_j(r) &= \sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right) \\
\Rightarrow f_j'(r) &= \sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)(\Delta t_j) \\
&= (\Delta t_j)\sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right),
\end{aligned}$$

so that

$$\begin{aligned}
\frac{f_j'(r)}{f_j(r)} &= \frac{(\Delta t_j)\sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)}{\sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j + \sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)} \\
&= \Delta t_j
\end{aligned}$$

and

$$\begin{aligned}\frac{\mathrm{d}\Pi_T(r)}{\mathrm{d}r} &= \Pi_0 \left(\prod_{j=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t_j + \sigma_i \epsilon_{i,j} \sqrt{\Delta t_j} \right) \right] \right) \left(\sum_{j=1}^{\lceil T/\tau \rceil} \frac{f_j'(r)}{f_j(r)} \right) \\ &= \Pi_T(r) \left(\sum_{j=1}^{\lceil T/\tau \rceil} \Delta t_j \right) \\ &= \Pi_T(r) T.\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{\mathrm{d}Y}{\mathrm{d}r} &= -Te^{-rT}[K - \Pi_T(r)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(r)\}}) \frac{\mathrm{d}\Pi_T(r)}{\mathrm{d}r} \\ &= -Te^{-rT}[K - \Pi_T(r)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(r)\}}) \Pi_T(r) T \\ &= Te^{-rT} \left(-[K - \Pi_T(r)]^+ + (-\mathbb{1}_{\{K>\Pi_T(r)\}}) \Pi_T(r) \right). \end{aligned} \tag{C.4}$$

C.4 $\frac{\mathrm{d}Y}{\mathrm{d}\rho}$

$$\begin{aligned}\frac{\mathrm{d}Y}{\mathrm{d}\rho} &= e^{-rT} \frac{\mathrm{d}[K - \Pi_T(\rho)]^+}{\mathrm{d}\Pi_T(\rho)} \frac{\mathrm{d}\Pi_T(\rho)}{\mathrm{d}\rho} \\ &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\rho)\}}) \frac{\mathrm{d}\Pi_T(\rho)}{\mathrm{d}\rho}.\end{aligned}$$

The second part of this derivative, $\frac{\mathrm{d}\Pi_T(\rho)}{\mathrm{d}\rho}$, is

$$\begin{aligned}\frac{\mathrm{d}\Pi_T(\rho)}{\mathrm{d}\rho} &= \frac{\mathrm{d}}{\mathrm{d}\rho}\Pi_T(\rho) \\ &= \frac{\mathrm{d}}{\mathrm{d}\rho}\Pi_0 \prod_{j=1}^{\lceil T/\tau \rceil} \left[\sum_{i=1}^{2} v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t_j + \sigma_i \epsilon_{i,j} \sqrt{\Delta t_j} \right) \right] \\ &= \Pi_0 \frac{\mathrm{d}}{\mathrm{d}\rho} \prod_{j=1}^{\lceil T/\tau \rceil} f_j(\rho).\end{aligned}$$

Next, the derivative of $f_j(\rho)$ needs to be calculated in order to apply Equation C.1, i.e.

$$f_j(\rho) = \sum_{i=1}^{2} \left(v_i \exp\left(\left(r - \frac{\sigma_i^2}{2} \right) \Delta t_j + \sigma_i \epsilon_{i,j} \sqrt{\Delta t_j} \right) \right)$$

$$
\begin{aligned}
&= \sum_{i=1}^{2}\left(v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\right) \\
&= v_1 \exp\left(\nu_1 \Delta t_j + \sigma_1 \epsilon_{1,j}\sqrt{\Delta t_j}\right) + v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2 \epsilon_{2,j}\sqrt{\Delta t_j}\right) \\
&= v_1 \exp\left(\nu_1 \Delta t_j + \sigma_1 Z_{1,j}\sqrt{\Delta t_j}\right) + v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2\left(Z_{2,j}\sqrt{1-\rho^2} + \rho Z_{1,j}\right)\sqrt{\Delta t_j}\right) \\
\Rightarrow f_j'(\rho) &= v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2\left(Z_{2,j}\sqrt{1-\rho^2} + \rho Z_{1,j}\right)\sqrt{\Delta t_j}\right)\left(\sigma_2\sqrt{\Delta t_j}\left(Z_{1,j} - \frac{\rho Z_{2,j}}{\sqrt{1-\rho^2}}\right)\right) \\
&= v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2 \epsilon_{2,j}\sqrt{\Delta t_j}\right)\left(\sigma_2\sqrt{\Delta t_j}\left(Z_{1,j} - \frac{\rho Z_{2,j}}{\sqrt{1-\rho^2}}\right)\right),
\end{aligned}
$$

so that

$$
\begin{aligned}
\frac{\mathrm{d}\Pi_T(\rho)}{\mathrm{d}\rho} &= \Pi_0\left(\prod_{j=1}^{\lceil T/\tau\rceil}\left[\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\right]\right)\left(\sum_{j=1}^{\lceil T/\tau\rceil}\frac{f_j'(\rho)}{f_j(\rho)}\right) \\
&= \Pi_T(\rho)\left(\sum_{j=1}^{\lceil T/\tau\rceil}\frac{v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2 \epsilon_{2,j}\sqrt{\Delta t_j}\right)\left(\sigma_2\sqrt{\Delta t_j}\left(Z_{1,j} - \frac{\rho Z_{2,j}}{\sqrt{1-\rho^2}}\right)\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)}\right).
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}\rho} &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(\rho)\}})\Pi_T(\rho) \times \\
&\quad\left(\sum_{j=1}^{\lceil T/\tau\rceil}\frac{v_2 \exp\left(\nu_2 \Delta t_j + \sigma_2 \epsilon_{2,j}\sqrt{\Delta t_j}\right)\left(\sigma_2\sqrt{\Delta t_j}\left(Z_{1,j} - \frac{\rho Z_{2,j}}{\sqrt{1-\rho^2}}\right)\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)}\right). \qquad \text{(C.5)}
\end{aligned}
$$

C.5 $\frac{\mathrm{d}Y}{\mathrm{d}T}$

It is important to note that the functions $\lceil x \rceil$ and $\lfloor x \rfloor$ are continuous around all $x \notin \mathbb{Z}$ and discontinuous at $x \in \mathbb{Z}$. Therefore because the function Y contains these functions, the derivative has to be split up into two sections, i.e. where T/τ is not an integer and where it is. Now,

$$
\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}T} &= [K - \Pi_T(T)]^+ \frac{\mathrm{d}}{\mathrm{d}T}(e^{-rT}) + e^{-rT}\frac{\mathrm{d}}{\mathrm{d}T}[K - \Pi_T(T)]^+ \\
&= -re^{-rT}[K - \Pi_T(T)]^+ + e^{-rT}\frac{\mathrm{d}}{\mathrm{d}T}[K - \Pi_T(T)]^+
\end{aligned}
$$

$$\begin{aligned} &= -re^{-rT}[K-\Pi_T(T)]^+ + e^{-rT}\frac{\mathrm{d}[K-\Pi_T(T)]^+}{\mathrm{d}\Pi_T(T)}\frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T} \\ &= -re^{-rT}[K-\Pi_T(T)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(T)\}})\frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T} \end{aligned} \tag{C.6}$$

for $T/\tau \in \mathbb{R}$. Then for $T/\tau \notin \mathbb{Z}$ the second part of this derivative, $\frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T}$, is

$$\begin{aligned} \frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T} &= \frac{\mathrm{d}}{\mathrm{d}T}\Pi_T(T) \\ &= \frac{\mathrm{d}}{\mathrm{d}T}\Pi_0 \prod_{j=1}^{\lceil T/\tau \rceil}\left[\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\right] \\ &= \Pi_0 \frac{\mathrm{d}}{\mathrm{d}T}\prod_{j=1}^{\lceil T/\tau \rceil} f_j(T). \end{aligned}$$

Next, the derivative of $f_j(r)$ needs to be calculated in order to apply Equation C.1. Note, however, that

$$f_j(T) = \begin{cases} \sum_{i=1}^{2} v_i \exp\left(\nu_i \tau + \sigma_i \epsilon_{i,j}\sqrt{\tau}\right), & j = 1, ..., \lfloor \frac{T}{\tau} \rfloor \\ \sum_{i=1}^{2} v_i \exp\left(\nu_i (T - \lfloor \frac{T}{\tau} \rfloor \tau) + \sigma_i \epsilon_{i,j}\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}\right), & j = \lceil \frac{T}{\tau} \rceil \end{cases}$$

so that

$$\Rightarrow f_j'(T) = \begin{cases} 0, & j = 1, ..., \lfloor \frac{T}{\tau} \rfloor \\ \sum_{i=1}^{2} v_i \exp\left(\nu_i (T - \lfloor \frac{T}{\tau} \rfloor \tau) + \sigma_i \epsilon_{i,j}\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}\right)\left(\nu_i + \frac{\sigma_i \epsilon_{i,j}}{2\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}}\right), & j = \lceil \frac{T}{\tau} \rceil . \end{cases}$$

Now,

$$\begin{aligned} \frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T} &= \Pi_0 \left(\prod_{j=1}^{\lceil T/\tau \rceil}\left[\sum_{i=1}^{2} v_i \exp\left(\nu_i \Delta t_j + \sigma_i \epsilon_{i,j}\sqrt{\Delta t_j}\right)\right]\right)\left(\sum_{j=1}^{\lceil T/\tau \rceil}\frac{f_j'(T)}{f_j(T)}\right) \\ &= \Pi_T(T)\frac{\sum_{i=1}^{2} v_i \exp\left(\nu_i (T - \lfloor \frac{T}{\tau} \rfloor \tau) + \sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil}\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}\right)\left(\nu_i + \frac{\sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil}}{2\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}}\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i (T - \lfloor \frac{T}{\tau} \rfloor \tau) + \sigma_i \epsilon_{i,\lceil \frac{T}{\tau} \rceil}\sqrt{T - \lfloor \frac{T}{\tau} \rfloor \tau}\right)}. \end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}T} &= -re^{-rT}[K-\Pi_T(T)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(T)\}})\frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T} \\
&= -re^{-rT}[K-\Pi_T(T)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(T)\}})\Pi_T(T)\times \\
&\quad \frac{\sum_{i=1}^{2} v_i \exp\left(\nu_i(T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau)+\sigma_i\epsilon_{i,\left\lceil\frac{T}{\tau}\right\rceil}\sqrt{T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau}\right)\left(\nu_i+\frac{\sigma_i\epsilon_{i,\left\lceil\frac{T}{\tau}\right\rceil}}{2\sqrt{T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau}}\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i(T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau)+\sigma_i\epsilon_{i,\left\lceil\frac{T}{\tau}\right\rceil}\sqrt{T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau}\right)} \\
&= -re^{-rT}[K-\Pi_T(T)]^+ + e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(T)\}})\Pi_T(T)\times \\
&\quad \frac{\sum_{i=1}^{2} v_i \exp\left(\nu_i(T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau)+\sigma_i\epsilon_{i,\left\lceil\frac{T}{\tau}\right\rceil}\sqrt{T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau}\right)\left(\nu_i+\frac{\sigma_i\epsilon_{i,\left\lceil\frac{T}{\tau}\right\rceil}}{2\sqrt{T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau}}\right)}{\sum_{i=1}^{2} v_i \exp\left(\nu_i(T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau)+\sigma_i\epsilon_{i,\left\lceil\frac{T}{\tau}\right\rceil}\sqrt{T-\left\lfloor\frac{T}{\tau}\right\rfloor\tau}\right)}.
\end{aligned} \tag{C.7}$$

The above equation is the $\frac{\mathrm{d}Y}{\mathrm{d}T}$ when $T/\tau \notin \mathbb{Z}$.

Therefore if $T/\tau \notin \mathbb{Z}$ then $\frac{\mathrm{d}\Pi_T(T)}{\mathrm{d}T}$ is Equation C.7, and when $T/\tau \in \mathbb{Z}$, this method cannot be used.

C.6 $\frac{\mathrm{d}^2Y}{\mathrm{d}\Pi_0^2}$

From the first section $\frac{\mathrm{d}Y}{\mathrm{d}\Pi_0}$ is

$$\begin{aligned}
\frac{\mathrm{d}Y}{\mathrm{d}\Pi_0} &= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(x)\}})\frac{\Pi_T(\Pi_0)}{\Pi_0} \\
&= e^{-rT}(-\mathbb{1}_{\{K>\Pi_T(x)\}})\prod_{j=1}^{\lceil T/\tau\rceil}\left[\sum_{i=1}^{2} v_i \exp\left(\left(r-\frac{\sigma_i^2}{2}\right)\Delta t_j+\sigma_i\epsilon_{i,j}\sqrt{\Delta t_j}\right)\right].
\end{aligned}$$

Therefore,

$$\frac{\mathrm{d}^2Y}{\mathrm{d}\Pi_0^2} = 0. \tag{C.8}$$

APPENDIX D

ALGORITHMS

Algorithm D.1 (MAIN())

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ, *type*
Output: *All scalars, vectors and matrices generated in this algorithm can be used in future computations.*
Algorithm:

if type $=$ *call then* $I \leftarrow -1$ *end if*
if type $=$ *put then* $I \leftarrow 1$ *end if*
$n_1 \leftarrow \lfloor 0.05 \times n \rfloor$
$n_2 \leftarrow n - n_1$
$\underline{P} \leftarrow \underline{0}$ *[vector of length* n_1*]*
$\underline{CV} \leftarrow \underline{0}$ *[vector of length* n_1*]*
if $(T \mod \tau = 0)$ *then*
  $\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau]$ *(length* T/τ*)*
else
  $\underline{\Delta t} \leftarrow [\tau, \tau, ..., \tau, T \bmod \tau]$ *(length* $\lceil T/\tau \rceil$*)*
end if
generate a $2\lceil T/\tau \rceil \times n_1$ *matrix containing the SS*
$Z_1, Z_2, \epsilon_1, \epsilon_2 \leftarrow 0$ $[\lceil T/\tau \rceil \times n_1$ *matrices*$]$
for $l = 1$ *to* n_1
  $\Pi \leftarrow \Pi_0$
  $CV_1 \leftarrow \Pi_0$
  $CV_2 \leftarrow \Pi_0$
  for $j = 1$ *to* $\lceil T/\tau \rceil$
    generate $\epsilon_1[j,l]$ *and* $\epsilon_2[j,l]$ *using the IPIT (Definition 2.1) and Equations 2.7 & 2.8 with* $Z_1[j,l]$ *and* $Z_2[j,l]$ *generated with the* l^{th} *column from the SS matrix*

$\Theta_1 \leftarrow \exp\left(\left(r - \frac{\sigma_1^2}{2}\right)\underline{\Delta t}[j] + \sigma_1\epsilon_1[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
$\Theta_2 \leftarrow \exp\left(\left(r - \frac{\sigma_2^2}{2}\right)\underline{\Delta t}[j] + \sigma_2\epsilon_2[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i\Theta_i$
$CV_1 \leftarrow CV_1 \times \Theta_1$
$CV_2 \leftarrow CV_2 \times \Theta_2$
end for
$\underline{P}[l] \leftarrow \max\{I \times (K - \Pi), 0\}e^{-rT}$
$\underline{CV}[l] \leftarrow \max\{I \times (K - CV_1), 0\}e^{-rT} + \max\{I \times (K - CV_2), 0\}e^{-rT}$
end for
$\bar{P} \leftarrow \sum_{i=1}^{n_1} \underline{P}[i]/n_1$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n_1}(\underline{P}[i] - \bar{P})^2/(n_1 - 1)}$
$\bar{CV} \leftarrow BS(S_0 = \Pi_0, T = T, r = r, \sigma = \sigma_1, K = K, type = type) +$
$BS(S_0 = \Pi_0, T = T, r = r, \sigma = \sigma_2, K = K, type = type)$
$\sigma_{P,CV} \leftarrow \sum_{i=1}^{n_1}(\underline{P}[i] - \bar{P})(\underline{CV}[i] - \bar{CV})/n_1$
$c^* \leftarrow -\sigma_{P,CV}/S_P^2$
$\underline{P} \leftarrow \underline{0}$ *[vector of length* n_2*]*
$\underline{P}_{CV} \leftarrow \underline{0}$ *[vector of length* n_2*]*
$\underline{CV} \leftarrow \underline{0}$ *[vector of length* n_2*]*
generate a $2\lceil T/\tau \rceil \times n_2$ *matrix containing the SS*
$Z_1, Z_2, \epsilon_1, \epsilon_2 \leftarrow 0$ $[\lceil T/\tau \rceil \times n_2$ *matrices*$]$
for $l = 1$ *to* n_2
$\Pi \leftarrow \Pi_0$
$CV_1 \leftarrow \Pi_0$
$CV_2 \leftarrow \Pi_0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
generate $\epsilon_1[j,l]$ *and* $\epsilon_2[j,l]$ *using the IPIT (Definition 2.1) and Equations 2.7 & 2.8 with* $Z_1[j,l]$ *and* $Z_2[j,l]$ *generated with the* l^{th} *column from the SS matrix*
$\Theta_1 \leftarrow \exp\left(\left(r - \frac{\sigma_1^2}{2}\right)\underline{\Delta t}[j] + \sigma_1\epsilon_1[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
$\Theta_2 \leftarrow \exp\left(\left(r - \frac{\sigma_2^2}{2}\right)\underline{\Delta t}[j] + \sigma_2\epsilon_2[j,l]\sqrt{\underline{\Delta t}[j]}\right)$
$\Pi \leftarrow \Pi \times \sum_{i=1}^{2} v_i\Theta_i$
$CV_1 \leftarrow CV_1 \times \Theta_1$
$CV_2 \leftarrow CV_2 \times \Theta_2$
end for
$\underline{\Pi}_T[l] \leftarrow \Pi$
$\underline{P}[l] \leftarrow \max\{I \times (K - \Pi, 0), 0\}e^{-rT}$
$\underline{CV}[l] \leftarrow \max\{I \times (K - CV_1), 0\}e^{-rT} + \max\{I \times (K - CV_2), 0\}e^{-rT}$
$\underline{P}_{CV}[l] \leftarrow \underline{P}[l] + c^*(\underline{CV}[l] - \bar{CV})$
if $type = call$ *then*

$\underline{\mathbb{1}}[l] \leftarrow ifelse(K < \underline{\Pi}_T[l], 1, 0)$
end if
if type = put then
$\underline{\mathbb{1}}[l] \leftarrow ifelse(K > \underline{\Pi}_T[l], -1, 0)$
end if
end for

■

Algorithm D.2 (Price of the ERBO)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ, *type*
Output: $\bar{O}_{CV}$ - *the price of the ERBO*
SE - *the SE of the estimate*
CI - *95% CI for* $\bar{O}_{CV}$
Algorithm:
MAIN(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ, *type)*
$\bar{O}_{CV} \leftarrow \sum_{i=1}^{n_2} \underline{P}_{CV}[i]/n_2$
$S_P \leftarrow \sqrt{\sum_{i=1}^{n_2}(\underline{P}_{CV}[i] - \bar{O}_{CV})^2/(n_2 - 1)}$
$SE \leftarrow S_P/\sqrt{n_2}$
$CI \leftarrow [\bar{O}_{CV} - 1.96 \times SE, \bar{O}_{CV} + 1.96 \times SE]$

Algorithm D.3 (Sensitivity of the ERBO with regard to Π_0)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ, *type*
Output: $\bar{G}$ - *sensitivity of the price of the ERBO with regard to* Π_0
SE - *the SE of the estimate*
CI - *95% CI for* $\bar{G}$
Algorithm:
MAIN(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*
for $i = 1$ *to* n
$\underline{G}[i] \leftarrow e^{-rT} \times \underline{\mathbb{1}}[i] \times \frac{\underline{\Pi}_T[i]}{\Pi_0}$
end for
$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n}(\underline{G}[i] - \bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$

Algorithm D.4 (Sensitivity of the ERBO with regard to σ_*)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ, *type*

Output: $\bar{G}$ - *sensitivity of the price of the ERBO with regard to* σ_*
SE - *the SE of the estimate*
CI - *95% CI for* $\bar{G}$
Algorithm:
$\underline{G} \leftarrow \underline{0}$
ALGA(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*
for $i = 1$ *to* n
$Temp_3 \leftarrow 0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
$Temp_1 \leftarrow X_*[j,i]\left(-\sigma_* \underline{\Delta t}[j] + \epsilon_*[j,i]\sqrt{\underline{\Delta t}[j]}\right)$
$Temp_2 \leftarrow X_1[j,i] + X_2[j,i]$
$Temp_3 \leftarrow Temp_3 + Temp_1/Temp_2$
end for
$\underline{G}[i] \leftarrow e^{-rT} \times \underline{\mathbb{1}}[i] \times \underline{\Pi}_T[i] \times Temp_3$
end for
$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 * SE, \bar{G} + 1.96 * SE]$

Algorithm D.5 (Sensitivity of the ERBO with regard to r)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{G}$ - *sensitivity of the price of the ERBO with regard to* r
SE - *the SE of the estimate*
CI - *95% CI for* $\bar{G}$
Algorithm:
$\underline{G} \leftarrow \underline{0}$
MAIN(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*
for $i = 1$ *to* n
$\underline{G}[i] \leftarrow -T \times \underline{P}[i] + e^{-rT} \underline{\mathbb{1}}[i] \times \underline{\Pi}_T[i] \times T$
end for
$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 * SE, \bar{G} + 1.96 * SE]$

Algorithm D.6 (Sensitivity of the ERBO with regard to ρ)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ

Output: $\bar{G}$ - *sensitivity of the price of the ERBO with respect to* ρ
SE - *the SE of the estimate*
CI - *95% CI for* $\bar{G}$
Algorithm:
$\underline{G} \leftarrow \underline{0}$
MAIN(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*
for $i = 1$ *to* n
$Temp_3 \leftarrow 0$
for $j = 1$ *to* $\lceil T/\tau \rceil$
$Temp_1 \leftarrow X_2[j,i] \times \left(\sigma_2 \sqrt{\underline{\Delta t}[j]} \left(Z_1[j,i] - \frac{\rho Z_2[j,i]}{\sqrt{1-\rho^2}} \right) \right)$
$Temp_2 \leftarrow X_1[j,i] + X_2[j,i]$
$Temp_3 \leftarrow Temp_3 + Temp_1/Temp_2$
end for
$\underline{G}[i] \leftarrow e^{-rT} \times \underline{\mathbb{1}}[i] \times \underline{\Pi}_T[i] \times Temp_3$
end for
$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$

Algorithm D.7 (Sensitivity of the ERBO with regard to T)

Input: n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ
Output: $\bar{G}$ - *sensitivity of the price of the ERBO with respect to* T
SE - *the SE of the estimate*
CI - *95% CI for* $\bar{G}$
Algorithm:
$\underline{G} \leftarrow \underline{0}$
MAIN(n, Π_0, τ, v_1, v_2, T, r, K, σ_1, σ_2, ρ*)*
if $T \mod \tau \neq 0$ *then*
for $i = 1$ *to* n
$Temp_1 \leftarrow X_1[\lceil \frac{T}{\tau} \rceil, i] \times \left(\left(r - \frac{\sigma_1^2}{2} \right) + \frac{\sigma_1 \epsilon_1[\lceil \frac{T}{\tau} \rceil, i]}{2\sqrt{\underline{\Delta t}[\lceil \frac{T}{\tau} \rceil]}} \right)$
$Temp_2 \leftarrow X_2[\lceil \frac{T}{\tau} \rceil, i] \times \left(\left(r - \frac{\sigma_2^2}{2} \right) + \frac{\sigma_2 \epsilon_2[\lceil \frac{T}{\tau} \rceil, i]}{2\sqrt{\underline{\Delta t}[\lceil \frac{T}{\tau} \rceil]}} \right)$
$Temp_3 \leftarrow X_1[\lceil \frac{T}{\tau} \rceil, i] + X_2[\lceil \frac{T}{\tau} \rceil, i]$
$\underline{G}[i] \leftarrow -r \times \underline{P}[i] + e^{-rT} \underline{\mathbb{1}}[i] \times \underline{\Pi}_T[i] \times (Temp_1 + Temp_2)/Temp_3$
end for
else
$h \leftarrow 0.1 \times T$

$\underline{k} \leftarrow [3, 1, -1, -3]$
$P_m \leftarrow 0$ *[*$0.1n \times 4$ *matrix]*
$\underline{G} \leftarrow \underline{0}$ *[vector of length* $0.1n$*]*
for $m = 1$ *to* 4
$T^* \leftarrow T$
$T \leftarrow T^* + \underline{k}[m] \times h$
$P_m[, m] \leftarrow MAIN\$P(n,\ \Pi_0,\ \tau,\ v_1,\ v_2,\ T,\ r,\ K,\ \sigma_1,\ \sigma_2,\ \rho)$
end for
$\underline{G}[j] \leftarrow (P_m[j, 1] - 3P_m[j, 2] + 3P_m[j, 3] - P_m[j, 4])/(2h)^3$ *for* $j = 1, ..., 0.1n$
$\bar{G} \leftarrow \sum_{i=1}^{0.1n} \underline{G}[i]/(0.1n)$
$\underline{G}[j] \leftarrow (P_m[j, 2] - P_m[j, 3])/(2h)$ *for* $j = 1, ..., 0.1n$
$\bar{G} \leftarrow \sum_{i=1}^{0.1n} \underline{G}[i]/(0.1n)$
$S_G \leftarrow \sqrt{\sum_{i=1}^{0.1n}(2h \times (\underline{G}[i] - \bar{G}))^2/(0.1n - 1)}$
$h_* \leftarrow \left(\frac{9S_G^2}{4G^2}\right)^{1/5}$
$P_m \leftarrow 0$ *[*$n \times 2$ *matrix]*
$\underline{G} \leftarrow \underline{0}$ *[vector of length* n*]*
$\underline{k} \leftarrow [1, -1]$
for $m = 1$ *to* 2
$T \leftarrow T^* + \underline{k}[m] \times h_*$
$P_m[, m] \leftarrow MAIN\$P(n,\ \Pi_0,\ \tau,\ v_1,\ v_2,\ T,\ r,\ K,\ \sigma_1,\ \sigma_2,\ \rho)$
end for
$\underline{G}[j] \leftarrow (P_m[j, 1] - P_m[j, 2])/(2h)$ *for* $j = 1, ..., n$
end if
$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$
$S_G \leftarrow \sqrt{\sum_{i=1}^{n}(\underline{G}[i] - \bar{G})^2/(n - 1)}$
$SE \leftarrow S_G/\sqrt{n}$
$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$

Algorithm D.8 (Gamma of the ERBO)

Input: $n,\ \Pi_0,\ \tau,\ v_1,\ v_2,\ T,\ r,\ K,\ \sigma_1,\ \sigma_2,\ \rho$
Output: $\bar{G}$ *- Gamma of the ERBO*
SE *- the SE of the estimate*
CI *- 95% CI for* $\bar{G}$
Algorithm:
$h \leftarrow 0.1 \times \Pi_0$
$\underline{k} \leftarrow [1, 0, -1]$
$P_m \leftarrow 0$ *[*$n \times 3$ *matrix]*
for $m = 1$ *to* 3
$T \leftarrow T^* + \underline{k}[m] \times h$

$P_m[,m] \leftarrow MAIN\$P(n,\ \Pi_0,\ \tau,\ v_1,\ v_2,\ T,\ r,\ K,\ \sigma_1,\ \sigma_2,\ \rho)$

end for

$\underline{G}[j] \leftarrow (P[j,1] - 2P[j,2] + P[j,3])/(h^2)$ *for* $j = 1, ..., n$

$\bar{G} \leftarrow \sum_{i=1}^{n} \underline{G}[i]/n$

$S_G \leftarrow \sqrt{\sum_{i=1}^{n} (\underline{G}[i] - \bar{G})^2/(n-1)}$

$SE \leftarrow S_G/\sqrt{n}$

$CI \leftarrow [\bar{G} - 1.96 \times SE, \bar{G} + 1.96 \times SE]$

All of the above algorithms can be coded in any mathematical software package. This was coded in *R* (R Development Core Team, 2010) and can be found in Appendix E.

APPENDIX E

R CODE FOR THE PRICE AND SENSITIVITIES OF THE ERBO

```
function (rho,sim,Sbele,T,rebal,v1,rf,K,sigma1,sigma2,type)
{
nsim <- sim
N1 <- round(sim*0.5)
N2 <- nsim - N1
nu1 <- rf - (sigma1^2)/2 #Used in Wiener Process
nu2 <- rf - (sigma2^2)/2 #Used in Wiener Process
v2 <- 1-v1          #v2 - Portion in Asset 2
seed<-sample(100000,1)

MAIN <- function(rho,sim,Sbele,T,rebal,v1,rf,K,sigma1,sigma2,type)
{
if (type=="put") I <- 1 else I <- -1
steps <- ceiling(T/rebal)
if(round(T%%rebal,2)==0)
  dt <- rep(rebal,T/rebal)
else
  dt <- c(rep(rebal,floor(T/rebal)),T%%rebal)
rands<- t(runif.sobol(N1,2*steps,scrambling=3,seed=seed))
firstRunif <- rands[1:(steps),]
secRunif <- rands[(steps+1):(2*steps),]
```

```
#prelim to find c*
Norm1 <- qnorm(firstRunif,0,1)
Norm2 <- qnorm(secRunif,0,1)
z1A <- Norm1
z2A <- rho*z1A + Norm2*sqrt(1-rho^2)

x1A <- v1*exp(nu1*dt+sigma1*z1A*sqrt(dt))
x2A <- v2*exp(nu2*dt+sigma2*z2A*sqrt(dt))
xsum <- x1A+x2A

if (T<=rebal)
  {
  cvx1A <- Sbele*x1A/v1
  cvx1A <- Sbele*x1A/v1
  PortA <- Sbele*xsum
  }
else
  {
  cvx1A <- Sbele*apply(x1A/v1,2,prod)
  cvx2A <- Sbele*apply(x2A/v2,2,prod)
  PortA <- Sbele*apply(xsum,2,prod)
  }

MST <- pmax(I*(K-PortA),0)

CV <- exp(-rf*T)*pmax(I*(K-cvx1A),0)+exp(-rf*T)*pmax(I*(K-cvx2A),0)
P<- exp(-rf*T)*MST
P.bar <- mean(P)
d11 <- (log(Sbele/K)+(rf+sigma1^2/2)*T)/(sigma1*sqrt(T)) #CV
d12 <- (log(Sbele/K)+(rf+sigma2^2/2)*T)/(sigma2*sqrt(T)) #CV
d21 <- d11-sigma1*sqrt(T) #CV
d22 <- d12-sigma2*sqrt(T) #CV
if(type=="put")
  {
  CV.bar1 <- K*exp(-rf*T)*pnorm(-d21)-Sbele*pnorm(-d11) #CV
  CV.bar2 <- K*exp(-rf*T)*pnorm(-d22)-Sbele*pnorm(-d12) #CV
  }
else
  {
  CV.bar1 <- Sbele*pnorm(d11) - K*exp(-rf*T)*pnorm(d21) #CV
```

```
  CV.bar2 <- Sbele*pnorm(d12)-K*exp(-rf*T)*pnorm(d22) #CV
  }
CV.bar <- CV.bar1 + CV.bar2 #CV
VarCV.hat <- sum((CV-CV.bar)^2)/(N1-1)
Cov.hat <- sum((CV-CV.bar)*(P-P.bar))/(N1-1)
c.star <- -Cov.hat/VarCV.hat

#second sims

rands<- t(runif.sobol(N2,2*steps,scrambling=3,seed=seed))
firstRunif <- rands[1:(steps),]
secRunif <- rands[(steps+1):(2*steps),]

Norm1 <- qnorm(firstRunif,0,1)
Norm2 <- qnorm(secRunif,0,1)
z1A <- Norm1
z2A <- rho*z1A + Norm2*sqrt(1-rho^2)

x1A <- v1*exp(nu1*dt+sigma1*z1A*sqrt(dt))
x2A <- v2*exp(nu2*dt+sigma2*z2A*sqrt(dt))
xsum <- x1A+x2A

if (T<=rebal)
  {
  cvx1A <- Sbele*x1A/v1
  cvx1A <- Sbele*x1A/v1
  PortA <- Sbele*xsum
  }
else
  {
  cvx1A <- Sbele*apply(x1A/v1,2,prod)
  cvx2A <- Sbele*apply(x2A/v2,2,prod)
  PortA <- Sbele*apply(xsum,2,prod)
  }

MSTA <- pmax(I*(K-PortA),0)
MST <- MSTA

CV2 <- exp(-rf*T)*pmax(I*(K-cvx1A),0)+exp(-rf*T)*pmax(I*(K-cvx2A),0)
P2<- exp(-rf*T)*MST
```

```
PCV <- P2+c.star*(CV2-CV.bar)
list(PCV=PCV,P2=P2,x1A=x1A,x2A=x2A,z1A=z1A,z2A=z2A,Norm1=Norm1,
  Norm2=Norm2,steps=steps,PortA=PortA,MSTA=MSTA,dt=dt)
}
results <- MAIN(rho,sim,Sbele,T,rebal,v1,rf,K,sigma1,sigma2,type)
PCV <- results$PCV
P2 <- results$P2
x1A <- results$x1A
x2A <- results$x2A
z1A <- results$z1A
z2A <- results$z2A
Norm1 <- results$Norm1
Norm2 <- results$Norm2
steps <- results$steps
PortA <- results$PortA
MSTA <- results$MSTA
dt <- results$dt

#PRICE

P.bar <- mean(PCV)
S <- sqrt(var(PCV)*N2/(N2-1))
PI <- P.bar - 1.96*S/sqrt(N2)
PI[2] <- P.bar + 1.96*S/sqrt(N2)
results.price <- c(P.bar,S/sqrt(N2),PI)

#SENS WRT PI_0
delta.DER<- PortA/Sbele
if (type=="put")
  delta.DERsims<-exp(-rf*T)*(-1)*(K>PortA)*delta.DER
else
  delta.DERsims<-exp(-rf*T)*(1)*(K<PortA)*delta.DER
bar.delta <- mean(delta.DERsims)
S.delta <- sqrt(var(delta.DERsims)*N2/(N2-1))
PI <- bar.delta - 1.96*S.delta/sqrt(N2)
PI[2] <- bar.delta + 1.96*S.delta/sqrt(N2)
results.delta <- c(bar.delta,S.delta/sqrt(N2),PI)

#SENS WRT SIG1
asset <- 1
```

```
sigs <- c(sigma1,sigma2)
if(asset==1)xin <- x1A else xin<-x2A
if(asset==1)zin <- z1A else zin<-z2A
veg.DER1 <- PortA
veg.DER2 <- xin*(-sigs[asset]*dt+sqrt(dt)*zin)
if (T<=rebal)
  veg.DER3 <- veg.DER1*veg.DER2/(x1A+x2A)
else
  veg.DER3 <- veg.DER1*apply(veg.DER2/(x1A+x2A),2,sum)
if (type=="put")
  veg.DERsims <- exp(-rf*T)*(-1)*(K>PortA)*veg.DER3
else
  veg.DERsims <- exp(-rf*T)*(1)*(K<PortA)*veg.DER3
bar.veg <- mean(veg.DERsims)
S.veg <- sqrt(var(veg.DERsims)*N2/(N2-1))
PI <- bar.veg - 1.96*S.veg/sqrt(N2)
PI[2] <- bar.veg + 1.96*S.veg/sqrt(N2)
results.veg1 <- c(bar.veg,S.veg/sqrt(N2),PI)

#SENS WRT SIG2
asset <- 2
sigs <- c(sigma1,sigma2)
if(asset==1)xin <- x1A else xin<-x2A
if(asset==1)zin <- z1A else zin<-z2A
veg.DER1 <- PortA
veg.DER2 <- xin*(-sigs[asset]*dt+sqrt(dt)*zin)
if (T<=rebal)
  veg.DER3 <- veg.DER1*veg.DER2/(x1A+x2A)
else
  veg.DER3 <- veg.DER1*apply(veg.DER2/(x1A+x2A),2,sum)
if (type=="put")
  veg.DERsims <- exp(-rf*T)*(-1)*(K>PortA)*veg.DER3
else
  veg.DERsims <- exp(-rf*T)*(1)*(K<PortA)*veg.DER3
bar.veg <- mean(veg.DERsims)
S.veg <- sqrt(var(veg.DERsims)*N2/(N2-1))
PI <- bar.veg - 1.96*S.veg/sqrt(N2)
PI[2] <- bar.veg + 1.96*S.veg/sqrt(N2)
results.veg2 <- c(bar.veg,S.veg/sqrt(N2),PI)
```

```
#SENS WRT r
if (type=="put")
  rho.DERsims <- T*exp(-rf*T)*(-MSTA+(-1)*(K>PortA)*PortA)
else
  rho.DERsims <- T*exp(-rf*T)*(-MSTA+(1)*(K<PortA)*PortA)
bar.rho <- mean(rho.DERsims)
S.rho<- sqrt(var(rho.DERsims)*N2/(N2-1))
PI <- bar.rho - 1.96*S.rho/sqrt(N2)
PI[2] <- bar.rho + 1.96*S.rho/sqrt(N2)
results.rho <- c(bar.rho,S.rho/sqrt(N2),PI)

#SENS WRT CORR
cor.DER1 <- PortA
cor.DER2 <- x2A*(sigma2*sqrt(dt)*(Norm1-((rho*Norm2)/sqrt(1-rho^2))))
if (T<=rebal)
  cor.DER3 <- cor.DER1* cor.DER2/(x1A+x2A)
else
  cor.DER3 <- cor.DER1*apply(cor.DER2/(x1A+x2A),2,sum)
if (type=="put")
  cor.DERsims <- exp(-rf*T)*(-1)*(K>PortA)*cor.DER3
else
  cor.DERsims <- exp(-rf*T)*(1)*(K<PortA)*cor.DER3
bar.cor <- mean(cor.DERsims)
S.cor<- sqrt(var(cor.DERsims)*N2/(N2-1))
PI <- bar.cor - 1.96*S.cor/sqrt(N2)
PI[2] <- bar.cor + 1.96*S.cor/sqrt(N2)
results.cor <- c(bar.cor,S.cor/sqrt(N2),PI)

#SENS WRT T
ind <- steps
if(round(T%%rebal,2) != 0)
  {
  thet.DER1 <- x1A[ind,]*(nu1+sigma1*z1A[ind,]*0.5*(T%%rebal)^(-1/2))+
    x2A[ind,]*(nu2+sigma2*z2A[ind,]*0.5*(T%%rebal)^(-1/2))
  thet.DER2 <- (x1A+x2A)[ind,]
  thet.DER3 <- thet.DER1/thet.DER2
  if(type=="put")
    thet.DERsims <- exp(-rf*T)*(-rf*MSTA+(-1)*(K>PortA)*PortA*thet.DER3)
  else
    thet.DERsims <- exp(-rf*T)*(-rf*MSTA+(1)*(K<PortA)*PortA*thet.DER3)
```

```
  }
else
  {
  h <- T*0.01
  hs <- c(3,1,-1,3)*h
  fh3 <-   MAIN(rho,sim=0.1*sim,Sbele,T+hs[1],rebal,v1,rf,K,sigma1,sigma2,type)$P2
  fh1 <-   MAIN(rho,sim=0.1*sim,Sbele,T+hs[2],rebal,v1,rf,K,sigma1,sigma2,type)$P2
  fhneg1 <-MAIN(rho,sim=0.1*sim,Sbele,T+hs[3],rebal,v1,rf,K,sigma1,sigma2,type)$P2
  fhneg3 <-MAIN(rho,sim=0.1*sim,Sbele,T+hs[4],rebal,v1,rf,K,sigma1,sigma2,type)$P2
  thirdder <- (fh3-3*fh1 + 3*fhneg1 - fhneg3)/((2*h)^3)
  firstder <- (fh1-fhneg1)/(2*h)
  var1stder <- var(firstder*h)
  bar.thirdder<- mean(thirdder)
  hstar <- ((9*var1stder)/(4*bar.thirdder^2))^(1/5)
  if(T-hstar < 0 ) hstar <- h
  h <- hstar
  hs <- h*c(1,-1)
  fh1 <- MAIN(rho,sim,Sbele,T+hs[1],rebal,v1,rf,K,sigma1,sigma2,type)$P2
  fhneg1 <- MAIN(rho,sim,Sbele,T+hs[2],rebal,v1,rf,K,sigma1,sigma2,type)$P2
  thet.DERsims <- (fh1-fhneg1)/(2*h)
  }
bar.thet <- mean(thet.DERsims)
S.thet <- sqrt(var(thet.DERsims)*N2/(N2-1))
PI <- bar.thet - 1.96*S.thet/sqrt(N2)
PI[2] <- bar.thet + 1.96*S.thet/sqrt(N2)
results.thet <- c(bar.thet,S.thet/sqrt(N2),PI)

#SECOND ORDER SENS WRT PI_0

h <- Sbele*0.01
hs <- c(1,0,-1)*h
fh1 <- MAIN(rho,sim,Sbele+hs[1],T,rebal,v1,rf,K,sigma1,sigma2,type)$P2
fh0 <- MAIN(rho,sim,Sbele+hs[2],T,rebal,v1,rf,K,sigma1,sigma2,type)$P2
fhneg1 <- MAIN(rho,sim,Sbele+hs[3],T,rebal,v1,rf,K,sigma1,sigma2,type)$P2
firstder <- (fh1 - 2*fh0 + fhneg1)/(h^2)
firstder.bar <- mean(firstder)
S <- sqrt(var(firstder)*sim/(N2-1))
PI <- firstder.bar - 1.96*S/sqrt(N2)
PI[2] <- firstder.bar + 1.96*S/sqrt(N2)
results.gamma  <- c(firstder.bar,S/sqrt(N2),PI)
```

```
round(rbind(results.price,results.delta,results.veg1,results.veg2,
  results.rho,results.cor,results.thet,results.gamma),4)
}
```

BIBLIOGRAPHY

Alexander, C. 2008a. *Market Risk Analysis I: Quantitative Methods in Finance*. 1st edition. England: John Wiley & Sons Ltd.

Alexander, C. 2008b. *Market Risk Analysis III: Pricing, Hedging and Trading Financial Instruments*. 1st edn. England: John Wiley & Sons Ltd.

Alexander, C. & Venkatramanan, A. 2009. *Analytic Approximations for Multi-Asset Option Pricing* [Online]. Available: http://ssrn.com/abstract=1424985 [2009, May 31]

Black, F. & Scholes, M. 1973. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3):637-654.

Boyle, P. 1977. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4:323-338.

Boyle, P., Broadie, M. & Glasserman, P. 1997. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, 21:1267-1321.

Brigo, D., Mercurio, F., Rapisarda, F. & Scotti, R. 2001. Approximated moment-matching dynamics for basket-options simulation [Online]. Available: http://ssrn.com/abstract=268433 [2010, June 1].

Carnell, R. 2009. LHS Edition 0.5 [Online]. Available: http://cran.r-project.org/web/packages/lhs/index.html [2010, June 1].

Chan, N.H. & Wong, H.Y. 2006. *Simulation Techniques in Financial Risk Management*. 1st Edition. New Jersey: John Wiley & Sons, Inc.

Curran, M. 1994. Valuing Asian and Portfolio Options by Conditioning on the Geometric Mean Price. *Management Science*, 40(12):1705-1711.

Deelstra, G., Diallo, I. and Vanmaele, M. 2008. Bounds for asian basket options. *Journal of Computational and Applied Mathematics*, 218:215-228.

Dhaene, J., Denuit, M., Goovaerts, M., Kaas, R. & Vyncke, D. 2002. The Concept of Comonotonicity in Actuarial Science and Finance: Applications. *Insurance: Mathematics & Economics*, 31(2):133-161.

Faure, H. 1982. Discrépence de suites associées à un système de numération (en dimension s). *Acta Arithmetica*, 41:337-351.

Fox, B.L. & Glynn, P.W. 1989. Replication schemes for limiting expectations. *Probability in the Engineering and Information Sciences*, 35:1297-1315.

Franco, J., Dupuy, D. and Roustant, O. 2010. DiceDesign Edition 1.0 [Online]. Available: http://cran.r-project.org/web/packages/DiceDesign/index.html [2010, June 1]

Frolov, A.S. & Chentsov, N.N. 1963. On the calculation of definite integrals dependent on a parameter by the Monte Carlo method. *USSR Journal of Computational Mathematics and Mathematical Physics*, 4:802-808.

Glasserman, P. 2004. *Monte Carlo Methods in Financial Engineering*. 1st Edition. New York: Springer Science + Business Media, LLC.

Glynn, P.W. 1989. Optimization of stochastic systems via simulation, in *Proceedings of the Winter Simulation Conference*. IEEE Press: 90-105.

Golub, G. & Van Loan, C.F. 1996. *Matrix Computations*. 3rd Edition. Baltimore: John Hopkins University Press.

Halton, J.H. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematic*, 2:84-90.

Hammersley, J.M. 1960. Monte Carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86:844-874.

Hull, J.C. 2009. *Options, Futures, and other Derivatives*. 7th Edition. USA: Pearson Education, Inc.

Huynh, H.T., Lai, V.S. & Soumaré, I. 2008. *Stochastic simulation and applications in finance with MATLAB programs*. 1st Edition. England: John Wiley & Sons.

Ju, N. 2002. Pricing Asian and basket options via Taylor expansion. *Journal of Computational Finance*, 5(3):79-103.

Krekel, M., De Kock, J., Korn, R. & Man, T.K. 2004. An analysis of pricing methods for basket options. *Wilmott Magazine*, 3:82-89.

Krykova, I. 2003. *Evaluating of Path-dependent Securities with Low Discrepancy Methods*, Worcester Polytechnic Institute.

Kwok, Y.K., Wong, H.Y. & Lau, K.W. 2001. Pricing Algorithms of Multivariate Path Dependent Options. *Journal of Complexity*, 17:773-794.

Mckay, M.D., Conover, W.J. and Beckman, R.J. 1979. A comparison of three methods for selecting input variables in the analysis of output from a computer code. *Technometrics*, 21:239-245.

Milevsky, M.A. and Posner, S.E. 1998a. A closed-form approximation for valuing basket options. *Journal of Derivatives*, 5(4):54-61.

Milevsky, M.A. and Posner, S.E. 1998b. Asian options, the sum of lognormals and the reciprocal gamma distribution. *The Journal of Financial and Quantitative Analysis*, 33(3):409-422.

Milevsky, M.A. and Posner, S.E. 1999. Valuing Exotic Options by Approximating the SPD with Higher Moments. *The Journal of Financial Engineering*, 7(2):109-125.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. 1st Edition. Philadelphia: Society for Industrial and Applied Mathematics.

Owen, A.B. 1998. Scrambling Sobol' and Niederreiter-Xing Points. *Journal of Complexity*, 14(4):466-489.

Prigent, J., Scaillet, O. & Renault, O.M. 2004. Option pricing with discrete rebalancing. *Journal of Empirical Finance*, 11:133-161.

R Development Core Team. 2010. R. Edition 2.11.1 [Online]. Available: http://www.r-project.org/ [2010, June 1].

Rice, J.A. 2007. *Mathematical Statistics and Data Analysis*. 3rd Edition. Belmont: Thomson Higher Education.

Sobol', I.M. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Journal of Computational Mathematics and Mathematical Physics*, 7:784-802.

Stein, M. 1987. Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29:143-151.

Tezuka, S. and Faure, H. 2003. I-binomial scrambling of digital nets and sequences. *Journal of Complexity*, 19:744-757.

Theron, N. 2007. *Aspects of some Exotic Options*, University of Stellenbosch.

Wang, X. 2001. Variance reduction techniques and Quasi-Monte Carlo methods. *Journal of Computational and Applied Mathematics*, 132(2):309-318.

Wilmott, P. 2007. *Quantitative Finance.* 2nd Edition. England: John Wiley & Sons Ltd.

Wuertz, D. 2010. fOptions. Edition 2110.78 [Online]. Available: http://cran.r-project.org/web/packages/fOptions/index.html [2010, June 1].

Zazanis, M.A. 1987. *Statistical Properties of Perturbation Analysis Estimates for Discrete Event Systems*, Harvard University.