Volume 28 (1), pp. 1–18
http://www.orssa.org.za

ORiON
ISSN 0529-191-X

Calculation aspects of the European rebalanced basket option using Monte Carlo methods: Valuation

CJ van der Merwe* WJ Conradie†

Received: 11 August 2011; Revised: 4 January 2012; Accepted: 12 April 2012

Abstract

Extra premiums may be charged to a client to guarantee a minimum payout of a contract on a portfolio that gets rebalanced back to fixed proportions on a regular basis. The valuation of this premium can be seen as that of the pricing of a European put option with underlying rebalanced portfolio. This paper finds the most efficient estimators for the value of this path-dependent multi-asset put option using different Monte Carlo methods. With the help of a refined method, computational time of the value decreased significantly. Furthermore, variance reduction techniques and Quasi-Monte Carlo methods delivered more accurate and faster converging estimates as well.

Key words: Simulation, stochastic programming, asset pricing, finance, insurance.

1 Introduction

A wide variety of products exist in life insurance and pension fund companies. Some of these products offer the holder of the product a minimum payout guarantee by charging them an extra premium. This extra guarantee forms a liability to the insurer that needs to be managed in terms of risks and must be valued daily. Due to the implementation of Solvency II across the European Union (for more information see Financial Services Authority, 2011) most nations have started adopting the same principles, with South Africa adopting the Solvency Assessment and Management (SAM) programme (Financial Services Board, 2010). According to Pillar I of these programmes, all Solvency Capital Requirements (SCR) need to be accurately measured, and kept as a reserve. A standard formula provided by the regulators may be used to help calculate the SCR, or an internal model may be used to estimate these requirements. Adopting an internal model

*Corresponding author: Department of Statistics and Actuarial Science, University of Stellenbosch, Private bag X1, Matieland, South Africa, 7602, email: carelvdmerwe@gmail.com

†Department of Statistics and Actuarial Science, University of Stellenbosch, Private bag X1, Matieland, South Africa, 7602.

brings forth the advantage of more accurate valuation and therefore, mostly a smaller SCR reserve.

The product considered in this paper is a portfolio, consisting of a assets, that is rebalanced back to fixed proportions, v_i, every τ years. Rebalancing is done by selling the better performing assets and buying the poorer performing assets. A minimum payout guarantee is offered to the client on this product and this forms the main focus of this paper.

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 European put option. This problem may therefore be seen as that of the valuation of a European put option with underlying Π.

In this paper the price (value) of this put option is estimated using different Monte Carlo (MC) methods in order to find the most efficient method. Due to the large and increasing computational power of corporations' clusters of servers, simulation becomes a feasible numerical method for calculating aspects of options where no closed-form solution or formulae exist — therefore the focus of this paper will only be 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 referred to as the European Rebalanced Basket Call/Put Option (ERBCO / ERBPO). Only the ERBPO will be considered, but the put-call parity for the European Rebalanced Basket Option (ERBO) is given in the concluding section and may be used to calculate the value of the ERBCO once the price of the ERBPO is found. Only a put option with an underlying portfolio that consists of two assets is considered in the results sections, but can easily be changed to that of an a-asset ERBO.

The focus of the next section is the valuation of the ERBPO using general MC methods. First, a simplistic method is used which is then substantially improved by using a new formula to simulate the value of the underlying portfolio. These are then compared in terms of computational time.

This is followed by a section which improves the error of the estimates with the help of Variance Reduction Techniques (VRTs). These methods are compared in terms of the estimates' standard error (SE) given a fixed simulation time. The fourth section improves the convergence of the estimates using Quasi-Monte Carlo (QMC) techniques, and the different methods are also compared in terms of different performance criteria.

2 General Monte Carlo

When used to value options, MC simulation uses the risk-neutral valuation result — the premium that needs to be charged for an option can be estimated by sampling paths for its underlying distribution to obtain the expected payoff in a risk-neutral world, and then discounting this payoff at the risk-free rate. The literature in this section is from Glasserman (2004, pp. 39–95) which builds on the work done by Boyle (1977).

A brief overview of the general MC Framework for option valuation will be provided in the following section. This will be followed by the derivation of the Simplistic MC (SMC) approach to valuing the ERBPO, which will be refined in the subsection that follows. This section will be concluded with a comparison of the two different methods in terms of value, error, and computational times.

2.1 Monte Carlo Framework for option valuation

Let X be a given random variable with $E[X] = \lambda$, where the true value is unknown, and $V(X) = \sigma^2$. In MC simulation, given the distribution of X, n independent observations of X, *i.e.* $\{X_i : i = 1, \ldots, n\}$, are generated. The parameter λ is estimated by $\hat{\lambda}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$, which is the sample mean of $\{X_1, \ldots, X_n\}$. This implies that $E[\hat{\lambda}(n)] = E[X] = \lambda$, hence $\hat{\lambda}(n)$ is an unbiased estimator of $E[X]$. Furthermore, $V(\hat{\lambda}(n)) = \sigma^2/n$. As the number of simulations, n, increases, $\hat{\lambda}(n)$ becomes a better estimator of λ as a consequence of the Law of Large Numbers (LLN) (Rice 2007, pp. 175).

The payoff at time T for the ERBPO is $\max\{K - \Pi_T, 0\}$. This needs to be discounted back to time $T = 0$ to obtain the value today. That is, in terms of the previous paragraph: $X = e^{-rT}\max\{K - \Pi_T, 0\}$ with $E[X] = \alpha$, the price of the option, and r the zero-coupon risk-free rate. It is important to note that throughout this paper it will be assumed that the term structure of risk-free rates is flat. It is, however, not difficult to incorporate a non-flat term structure of interest rates, as one simply need to calculate the forward rates, $r_{t,t+\Delta} = ((t+\Delta) \times r_{t+\Delta} - t \times r_t)/\Delta$ (with Δ the length of the time step being simulated) when simulating between time steps.

Therefore, the problem changes to simulating the variate Π_T. Once this has been simulated, the payoff can easily be discounted. The simulation of Π_T forms the focus of the next two subsections.

2.2 Simplistic Monte Carlo

Before considering the simulation of an a-asset option, the process followed by the underlying stocks needs to be discussed. Each stock has a continuous dividend yield, q_j, with $j = 1, \ldots, a$. For multi-asset options the correlated underlying stocks are assumed to follow the Geometric Brownian motion process (considering that in risk-neutral valuation all assets are assumed to have the same return, r) resulting in

$$\frac{\mathrm{d}S_j}{S_j} = (r - q_j)\,\mathrm{d}t + \sigma_j \mathrm{d}z_j,$$

with $\hat{E}[\mathrm{d}z_j \mathrm{d}z_k] = \rho_{jk}\,\mathrm{d}t$, $\hat{E}$ the expected value in the risk-neutral world, and ρ_{jk} the correlation between assets j and k. In sampling the paths of these assets (for $j = 1, \ldots, a$ with correlation matrix Σ), the following well-known result

$$S_{j,t+\Delta t} = S_{j,t} \exp\left[\left(r - q_j - \frac{\sigma_j^2}{2}\right)\Delta t + \sigma_j \epsilon_j \sqrt{\Delta t}\right], \quad \text{for } j = 1, \ldots, a, \quad \text{with}$$

$$\begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_a \end{bmatrix} \sim MVN_a\left(\underline{0}; \Sigma\right) \quad \text{and} \quad \Sigma = \begin{bmatrix} 1 & \rho_{1,2} & \cdots & \rho_{1,a} \\ \rho_{2,1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \rho_{a,1} & \cdots & \cdots & 1 \end{bmatrix}$$

is obtained. This can be simulated with the use of Cholesky factorization. It will be explained in terms of generating a correlated normally distributed variables ϵ_1, ϵ_2,...,ϵ_a. A sequence of a uncorrelated normally distributed variables $Z_1, Z_2, \ldots, Z_a$ can be generated and transformed with $\underline{\epsilon} = M\underline{Z}$, where $\underline{\epsilon}^T = (\epsilon_1, \ldots, \epsilon_a)$ and $\underline{Z}^T = (Z_1, \ldots, Z_a)$ are column vectors. The matrix $M : a \times a$ must satisfy $MM^T = \Sigma$, with $\Sigma : a \times a$ the correlation matrix. This can be confirmed by taking the expectation of $\underline{\epsilon}\,\underline{\epsilon}^T = M\underline{Z}\,\underline{Z}^T M^T$ as $E[\underline{\epsilon}\,\underline{\epsilon}^T] = ME[\underline{Z}\,\underline{Z}^T]M^T = MM^T = \Sigma$.

Therefore, to simulate paths for the underlying stocks, only a uncorrelated $N(0,1)$ variables are needed. These are simulated by generating $\underline{U} \sim U(0,1)^a$ and applying the Inverse Probability Integral Transform (IPIT) (Rice 2007, pp. 352–358). It is important to note that the whole simulation process originates from $\underline{U} \sim U(0,1)^a$. Note, however, that the dimensionality increases as more time steps are simulated: say 5 jumps for each asset needs to be simulated and there are 4 assets, then the dimensionality of one simulation (that is the portfolio value at time T) depends on 4 independent $\underline{U} \sim U(0,1)^5$, or simply $\underline{U} \sim U(0,1)^{5\cdot 4=20}$.

The portfolio Π gets rebalanced every τ years, with the option expiring at T. The underlying assets need to be simulated on times $\{t = \ell\tau : \ell = 1, \ldots, \lfloor T/\tau \rfloor, T/\tau\}$ which implies that the number of jumps for each stock that need to be simulated, is $\lceil T/\tau \rceil$. Thus, the dimension of the problem changes to $a \cdot \lceil T/\tau \rceil$ with a the number of assets underlying the portfolio.

To simulate the value of the portfolio (in order to calculate the discounted payoff) the following needs to be considered first: the value of the portfolio, at any given time t, is expressed as $\Pi_t = w_{1,\ell\tau}S_{1,t} + \ldots + w_{a,\ell\tau}S_{a,t}$, with ℓ the timestamp of the rebalancing prior to time t and $w_{j,\ell\tau}$ the number of units of asset j held in Π at that point in time. Further, note that $\Pi_{\ell\tau-\delta t} = \Pi_{\ell\tau}$ when $\delta t \to 0$ — that is, the value of the portfolio before rebalancing is exactly the same as after rebalancing.

By using the jumps of the underlying stocks $\{S_{j,\ell\tau} : j = 1, \ldots, a; \ell = 1, \ldots, \lfloor T/\tau \rfloor, T/\tau\}$ and the fact that $\Pi_{\ell\tau-\delta t} = \Pi_{\ell\tau}$ for $\delta t \to 0$ we may use

$$w_{j,\ell\tau} = \frac{v_j \Pi_{\ell\tau-\delta t}}{S_{j,\ell\tau-\delta t}} = \frac{v_j \Pi_{\ell\tau}}{S_{j,\ell\tau}}, \tag{1}$$

with $\ell = 0, 1, \ldots, \left\lfloor \frac{T}{\tau} \right\rfloor$ and $j = 1, \ldots, a$, and v_j the proportion of the portfolio value invested in asset j to calculate $\{w_{j,\ell\tau} : j = 1, \ldots, a; \ell = 1, \ldots, \lfloor T/\tau \rfloor\}$.

Values for $\{w_{j,\lfloor T/\tau \rfloor\tau} : j = 1, \ldots, a\}$ are found by means of equation (1) and a possible value

$$\Pi_T = \sum_{j=1}^{a} \left(w_{j,(\lfloor T/\tau \rfloor)\tau} S_{j,T} \right)$$

can be simulated. This will be used to simulate n independent values of X, $\{X_i : i = 1, \ldots, n\}$, such that the estimator for the price of the ERBPO is given by

$$\begin{aligned} \hat{\alpha}_{SMC} &= \frac{1}{n}\sum_{i=1}^{n} X_i \\ &= \frac{1}{n}\sum_{i=1}^{n} e^{-rT}\max\{K - \Pi_{i,T}, 0\}. \end{aligned}$$

2.3 Refined Monte Carlo

It can be proved by means of mathematical induction that the value of Π_T that consists of two non-dividend paying assets, with $\underline{\Delta t}^T = (\tau, \ldots, \tau, T \mod \tau)$, can be calculated with

$$\Pi_T = \Pi_0 \times \prod_{\ell=1}^{\lceil T/\tau \rceil}\left[\sum_{j=1}^{2} v_j \exp\left(\left(r - \frac{\sigma_j^2}{2}\right)\Delta t_\ell + \sigma_j \epsilon_j \sqrt{\Delta t_\ell}\right)\right]. \tag{2}$$

For a portfolio that consists of a dividend paying assets, with dividend yield q_i, (2) can be generalised to

$$\Pi_T = \Pi_0 \times \prod_{\ell=1}^{\lceil T/\tau \rceil}\left[\sum_{j=1}^{a} v_j \exp\left(\left(r - q_j - \frac{\sigma_j^2}{2}\right)\Delta t_\ell + \sigma_j \epsilon_j \sqrt{\Delta t_\ell}\right)\right].$$

This formula may be seen as the initial portfolio value, Π_0, growing proportionally to the growth rates of the different underlying correlated assets.

Hence, to simulate the price of this portfolio one only needs an observed value of $\underline{U} \sim U(0,1)^d$, with $d = a \cdot \lceil T/\tau \rceil$, to obtain $\lceil T/\tau \rceil$ correlated normally distributed $\underline{\epsilon}$ vectors, where $\underline{\epsilon} \sim MVN_a(\underline{0}; \Sigma)$. This method delivers exactly the same results, but in considerably less computational time.

The final Refined Monte Carlo (RMC) estimator is given by

$$\begin{aligned} \hat{\alpha}_{RMC} &= \frac{1}{n}\sum_{i=1}^{n} X_i \\ &= \frac{1}{n}\sum_{i=1}^{n} e^{-rT}\max\{K - \Pi_{i,T}, 0\}, \end{aligned}$$

with

$$\Pi_{i,T} = \Pi_0 \times \prod_{\ell=1}^{\lceil T/\tau \rceil}\left[\sum_{j=1}^{a} v_j \exp\left(\left(r - q_j - \frac{\sigma_j^2}{2}\right)\Delta t_\ell + \sigma_j \epsilon_{i,j} \sqrt{\Delta t_\ell}\right)\right],$$

for $n \cdot \lceil T/\tau \rceil$ independently identically distributed (i.i.d.) $\underline{\epsilon_i} \sim MVN_a(\underline{0}; \Sigma)$. The $\underline{\epsilon}$ vectors are obtained by generating $\underline{U_i} \sim U(0,1)^a$, using the IPIT to obtain $\underline{Z_i} \sim MVN_a(\underline{0}, I)$, and finally transforming it to $\underline{\epsilon_i} = M\underline{Z_i}$.

`R` uses the function `Random` in its `base` package (R Development Core Team and contributors worldwide 2011) as the Random Number Generator (RNG). This function randomly chooses which algorithm to use to generate the random uniformly distributed variables. The seed consists of a vector of different integers, and the length depends on the methods chosen, and would therefore be cumbersome to include here. Note that, whenever simulations are performed for a specific result, `R` enables the user to keep that initial seed constant, and this was done throughout all simulations.

2.4 Results for the comparison of the simplistic vs. the refined Monte Carlo approach

This section provides a comparison of the SMC to the RMC approach. Simulations were performed over 9 combinations of $\rho \in \{\text{-}0.5,\ 0,\ 0.5\}$, $\Pi_0 \in \{500,\ 1\,000,\ 1\,500\}$ and $n \in \{500,\ 50\,000,\ 500\,000\}$, for a two-asset ERBPO. The other parameters were held fixed as follows: $\tau = 1$, $v_1 = v_2 = 0.5$, $T = 10$, $S_{1,0} = 15$, $S_{2,0} = 20$, $r = 0.03$, $\sigma_1 = \sigma_2 = 0.3$, $q_1 = q_2 = 0$ and $K = 1\,000$. The initial seed for the RNG was also fixed so that results from different methods could be compared. The Price and SE were found to be exactly the same for both approaches with the only difference being the computational time for the two methods.

Note that, the combinations of the inputs used were chosen arbitrarily — any other parameter could have been chosen as part of the different combinations. The most important parameter that is incorporated here is the number of simulations, n, this would significantly increase the computational time for the different combinations, while the other inputs simply group each simulation.

Figure 1 shows the different methods in terms of computational time on a logarithmic scale, with distinction made between $n \in \{500,\ 50\,000,\ 500\,000\}$. Note that these differences will increase/decrease as the dimensionality of the problem increases/decreases.

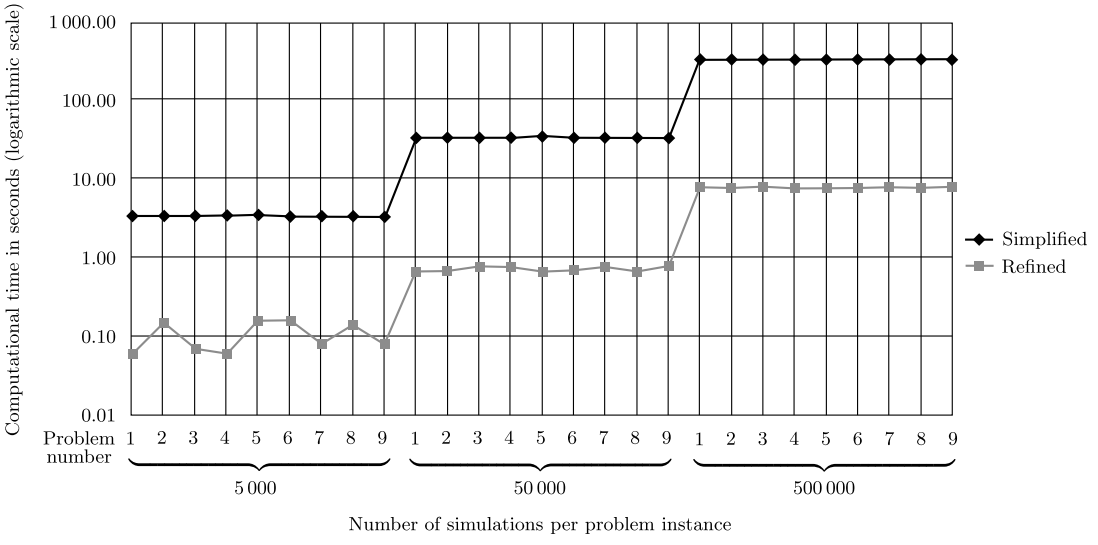


Figure 1: *A graph of the computational times for the simplistic and refined MC methods for three different number of simulations.*

The computational time is considerably less for the refined method. It is interesting to note that, in nominal terms, the computational times are considerably reduced. For $n = 500\,000$ the simplistic approach took approximately 340 seconds, where the refined method only took approximately 8 seconds yielding a reduction of approximately $5\frac{1}{2}$ minutes. In real terms, the refined method only took approximately 2% of the computational time of the simplistic method resulting in approximately a 98% reduction. The decrease in computational time becomes especially important in practice when valuing a large number of contracts at the same time.

3 Variance reduction techniques

In this section three different VRTs are applied to the RMC simulation of the ERBPO. All the methodology behind each VRT in terms of the ERBPO will briefly be described in each subsection, after which the estimator of the ERBPO will be given. The section concludes with a comparison of results, facilitating a choice with regard to the superior method. A brief discussion of the methodology behind VRTs is supplied first.

In MC simulation, $\lambda = E[X]$ is estimated by generating a sample $\{X_i : i = 1, \ldots, n\}$ and then determining $\widehat{\lambda}(n) = \frac{1}{n}\sum_{i=1}^{n} X_i$. Furthermore, the SE of the estimator is $\sigma/\sqrt{n}$ with σ^2 the variance of X. Note that there are two elements that influence the SE, namely $\sqrt{n}$ and σ. The first element can easily be interpreted: the more simulations that are performed, 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.

The SE is directly related to the width of the confidence intervals (CIs) constructed after simulations are performed. If the SE can be reduced, then smaller CIs can be obtained. In this section methods are thus introduced to reduce the size of σ. The best method will be chosen on the basis of a fixed computational time and size of the SE. That is, given a fixed amount of time, which method yields the smallest SE and therefore the smallest CI? Some VRTs increased the computational time of the simulation, but this was brought into consideration since simulations across all VRTs where performed over a fixed amount of time. A VRT that increases computational time will thus be simulated a smaller number of times.

3.1 Antithetic variables

Antithetic variables (AVs) attempt to reduce the variability of the simulations by producing a set of simulations with the help of uniformly distributed random variables and then a second set of simulations are performed with a set of perfectly negative correlated uniformly distributed variables to the first set.

Let $X = H(\underline{U}) = e^{-rT}\max\{K - \Pi_{U,T}, 0\}$ and $Y = H(\underline{V}) = e^{-rT}\max\{K - \Pi_{V,T}, 0\}$ with $\underline{U} \sim U(0,1)^{a\cdot\lceil T/\tau\rceil}$ and $\underline{V} = (1 - \underline{U}) \sim U(0,1)^{a\cdot\lceil T/\tau\rceil}$. Then $E[X] = E[Y] = \alpha$ and $V(X) = V(Y) = \sigma^2$.

Note that since, theoretically, $Cov(U, V) = -1/12$, it implies that $Corr(U, V) = -1$.

Thus, using these in a monotonic function, $H(\cdot)$ would cause

$$Cov(H(\underline{U}), H(\underline{V})) < 0.$$

Therefore, let $X_{AV} = \frac{X+Y}{2}$ with

$$E[X_{AV}] = \alpha$$

and

$$V(X_{AV}) = \sigma^2(1 + \rho_{X,Y}).$$

Generate $n/2$ observations to obtain the average, that is

$$\begin{aligned} \hat{\alpha}_{AV} &= \frac{1}{n/2}\sum_{i=1}^{n/2} X_{AV,i} \\ &= \frac{1}{n/2}\sum_{i=1}^{n/2}\left(\frac{X_i + Y_i}{2}\right) \\ &= \frac{1}{n/2}\sum_{i=1}^{n/2}\left(\frac{H(\underline{U}_i) + H(\underline{V}_i)}{2}\right), \end{aligned}$$

with X_i and Y_i (i.i.d.) simulations of X and Y as given above (using negatively correlated underlying uniformly distributed variables). Hence

$$E[\hat{\alpha}_{AV}] = \alpha$$

and

$$V(\hat{\alpha}_{AV}) = \frac{\sigma^2}{n}(1 + \rho_{X,Y}).$$

It is clear that the SE reduces when $\rho_{X,Y} < 0$.

3.2 Control variates

Control Variates (CVs) incorporates a variate into the simulation process, of which the true value is known. This subsection will start by mathematically explaining why this could possibly reduce the variablility of the simulated variables. A more detailed summary of CVs may be found in Chan and Wong (2006, pp. 104–109) and will be applied here to the pricing of the ERBPO.

When $\alpha = E[X]$ is estimated with MC simulation a CV, Y, may 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 of α, *i.e.* $E[X_{CV}] = \alpha$. By taking the derivative of $V(X_{CV}) = \sigma_X^2 + c^2\sigma_Y^2 + 2c\sigma_{X,Y}$ and setting it equal to 0, the optimal c, called c^*, can be found as $c^* = -\sigma_{X,Y}/\sigma_Y^2$. This c^* is estimated as $\hat{c}^* = -\hat{\sigma}_{X,Y}/\hat{\sigma}_Y^2$, with $\hat{\sigma}_{X,Y}$ the sample covariance and $\hat{\sigma}_Y^2$ the sample variance.

The CV, Y, will be chosen as a basket of a options, each being a plain vanilla European put option (this changes to call options when working with the ERBCO) on asset j, with $j = 1, \ldots, a$. That is $Y_{CV} = \sum_{j=1}^{a} Y_j$ with Y_j the discounted payoff from each of these put options. For simplicity, all assets are assumed to have an initial value of Π_0 such that

$$Y_i = e^{-rT} \max\{K - S_{j,T}, 0\},$$

with $S_{j,T}$ the value of the single dividend paying stock at time T.

The true value for each of these a options can be found with the help of the Black-Scholes (BS) option pricing formulae (Black and Scholes 1973) since all volatility surfaces for each of the underlying asset are known. The true value is denoted by $\mu_{Y_j} = E[Y_j]$ such that $\mu_{CV} = \sum_{i=j}^{a} \mu_{Y_j}$. The observations of the CVs, Y_j, are computed from the already simulated observations of Π_T by splitting the calculation of Π_T.

If

$$\Pi_T = \Pi_0 \times \prod_{\ell=1}^{\lceil T/\tau \rceil} \left[\sum_{j=1}^{a} v_j \exp\left(\left(r - q_j - \frac{\sigma_j^2}{2} \right) \Delta t_\ell + \sigma_j \epsilon_j \sqrt{\Delta t_\ell} \right) \right]$$

then let

$$\Theta_{j,\ell} = \exp\left(\left(r - q_j - \frac{\sigma_j^2}{2} \right) \Delta t_\ell + \sigma_j \epsilon_j \sqrt{\Delta t_\ell} \right),$$

with $j = 1, \ldots, a$ and $\ell = 1, \ldots, \lceil T/\tau \rceil$, such that

$$\Pi_T = \Pi_0 \times \prod_{\ell=1}^{\lceil T/\tau \rceil} \left[\sum_{j=1}^{a} v_j \Theta_{j,\ell} \right].$$

Using $\Theta_{j,\ell}$, the prices of dividend paying assets with initial prices, Π_0, is easily computed with

$$S_{j,T} = \Pi_0 \times \prod_{\ell=1}^{\lceil T/\tau \rceil} \Theta_{j,\ell}, \quad j = 1, \ldots, a,$$

such that $S_{j,T}$, with $j = 1, \ldots, a$, may be used in the simulation of the prices of the vanilla puts Y_j. Hence the true values are calculated with $\mu_{Y_j} = Ke^{-rT}\Phi(-d_{2,j}) - \Pi_0\Phi(-d_{1,j})$ where

$$d_{1,j} = \frac{\ln\left(\frac{\Pi_0}{K}\right) + \left(r - q_j + \frac{\sigma_j^2}{2} \right) T}{\sigma_j \sqrt{T}}, \qquad d_{2,j} = d_{1,j} - \sigma_j \sqrt{T}$$

and $\Phi(\cdot)$ is the cumulative standard normal distribution function (Black and Scholes 1973).

The final estimator of the price of the ERBPO using CVs is given by

$$\hat{\alpha}_{CV} = \frac{1}{n} \sum_{i=1}^{n} \left(X_i + \hat{c}^* \left(Y_i - \mu_{CV} \right) \right),$$

with X_j as i.i.d. observations of the discounted simulated payoffs of the ERBPO, Y_j as i.i.d. observations of $Y_i = e^{-rT}\max\{K - S_{j,T}, 0\}$, μ_{CV} as the true value of the sum of the a put options found using the BS pricing formulae, and

$$\hat{c}^* = -\frac{\frac{1}{n-1}\sum_{k=1}^{n}\left(X_k - \bar{X}\right)\left(Y_k - \bar{Y}\right)}{\frac{1}{n-1}\sum_{k=1}^{n}\left(Y_k - \bar{Y}\right)^2},$$

with $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$, the sample averages of the simulated observations.

3.3 Latin hypercube sampling

Latin Hypercube Sampling (LHS) is the method of systematic sampling for higher dimensions. It was first introduced by McKay, Conover, and Beckman (1979) and further analysed in Stein (1987). 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. The method helps with the reduction of variance by sampling systematically (evenly spread out) throughout the unit hypercube.

The package `lhs` (Carnell 2009) for the statistical program `R` (R Development Core Team 2009), has a built-in function for generating an LHS of size B for dimension d. This was used to generate the underlying $\underline{\mathrm{U}} \sim U(0,1)^d$ for the simulations.

If n samples are generated 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 being that the LHS samples are chosen evenly through the unit hypercube. Therefore, the SE of this method's estimate will be almost exactly the same as for the normal refined MC method. No variance reduction will be visible.

Glasserman (2004, p. 242) suggests generating i.i.d. estimators $\hat{\alpha}_1(B), \ldots, \hat{\alpha}_m(B)$, each based on an LHS of size B, such that the estimator, with $n = m \cdot B$, becomes

$$\begin{aligned}\hat{\alpha}_{LHS} &= \frac{1}{m}\sum_{i=1}^{m}\hat{\alpha}_i(B) \\ &= \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{B}\sum_{k=1}^{B} Y_{i,k}\right),\end{aligned}$$

with $\{Y_{i,k}\}$ generated from B uniformly distributed variables on $(0,1)^d$, generated using m different LHSs. Note that $E[\hat{\alpha}_{LHS}] = \alpha$, and that the SE of this estimator is the standard deviation of $\hat{\alpha}_1(B), \ldots, \hat{\alpha}_m(B)$ divided by $\sqrt{m}$.

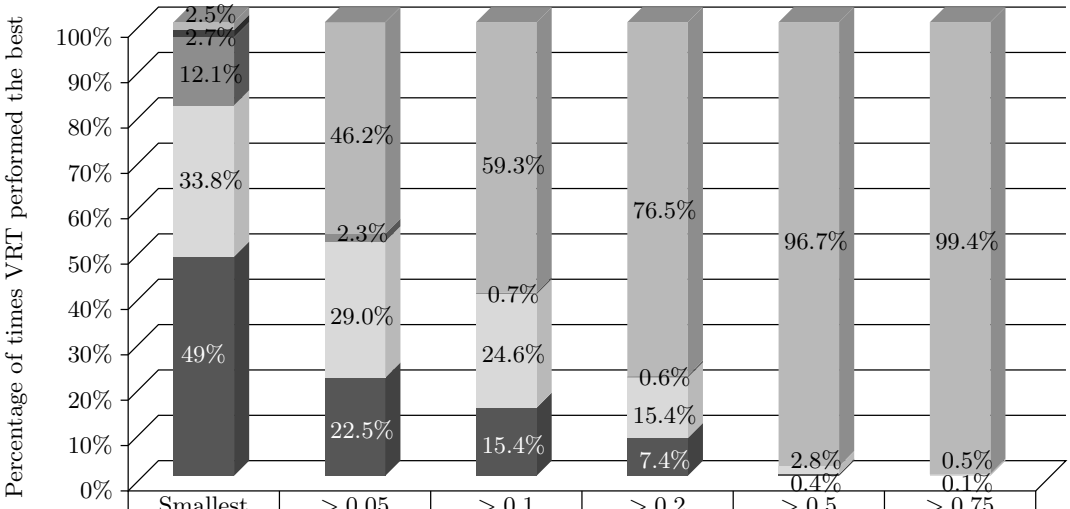
3.4 Results for measuring the efficiency of different VRTs

To determine the efficiency of VRTs, the various methods were compared with each other. In total, 1 620 two-asset ERBPO problem instances were constructed through all combina-

tions 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 $r \in \{0.01,\ 0.03,\ 0.08\}$; the standard deviation of the first asset $\sigma_1 \in \{0.05,\ 0.3\}$; the standard deviation of the second asset $\sigma_2 = 0.3$; and strike price $K = 1\,000$. Each of these inputs used in the combinations affect the price of the ERBO. As an example, if the option were far out-of-the-money ($\Pi_0 >> K$) most of the simulations will return a result of 0, and therefore would not return a significant SE. Furthermore, no variance reduction effect will be visible.

The ERBPO was priced using the RMC (Normal) method as well as the RMC 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 one second. That is, given one second, which method will yield, relative to the other methods, the smallest SE and therefore the smallest CI? Due to the extent of the results, only a summary is provided here.

Figure 2 gives the percentage of times a certain VRT outperformed others, *i.e.* produced the smallest SE in one second. The second column (Smallest) counts the percentage of times a VRT outperformed other methods. The NA (Not Available) row indicates the situation where there was no clear winner. The next five columns indicate the percentage of times a certain VRT outperformed the VRT that performed second best relative to a certain threshold. These thresholds are 0.05, 0.1, 0.2, 0.5 and 0.75. For example, in 29% of the 1 620 instances the CV method produced an SE that was more than 0.05 smaller than the second best method.



	Smallest	> 0.05	> 0.1	> 0.2	> 0.5	> 0.75
NA	2.5%	46.2%	59.3%	76.5%	96.7%	99.4%
Normal	2.7%	0.0%	0.0%	0.0%	0.0%	0.0%
LHS	12.1%	2.3%	0.7%	0.6%	0.0%	0.0%
CV	33.8%	29.0%	24.6%	15.4%	2.8%	0.5%
AV	49%	22.5%	15.4%	7.4%	0.4%	0.1%

Difference between best and second best method's standard error

Figure 2: *Graphical representation of comparison of VRTs over 1 620 problem instances.*

Initially, 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 number of times AV performed the best reduces much faster than that of CV, when looking at the margin by which it outperformed the second most efficient VRT. CVs will be chosen as the better VRT. Therefore, when applying this VRT, the SE of the estimate should mostly be smaller than for other methods.

In the next section, QMC techniques will be used to price the options.

4 Quasi-Monte Carlo techniques

In this section the application of QMC techniques to the normal RMC method are considered. QMC techniques attempt to accelerate convergence of the method by using low discrepancy sequences (LDSs) which are deterministically chosen sequences. The methodology behind the implementation of QMC in the ERBPO problem is first discussed, followed by a method to evaluate the performance of these methods — one with fixed dimensionality and the other over increasing dimensionality.

4.1 LDS and RLDS

Several QMC techniques exist that are implemented in the pricing of the ERBO. These may be split up into two different sequences, namely normal LDSs and randomised LDSs (RLDSs). Randomising QMC points opens the possibility of measuring error through a CI while preserving much of the accuracy of pure QMC. Randomised QMC (RQMC), which uses RLDSs, thus seeks to combine the best feature of ordinary MC and QMC. Randomisation sometimes improves accuracy as well.

The three LDSs that were implemented was the Halton (HS), Faure (FS) and Sobol' (SS) sequences (Halton 1960, Hammersley 1960, Faure 1982, Sobol' 1967). The algorithm used for pricing the ERBPO was exactly the same as for the RMC method except that LDSs were used as the $\underline{U} \sim U(0,1)^d$, with $d = a \cdot \lceil T/\tau \rceil$, instead of generating them with a RNG. Four RLDSs were also implemented, all based on the SS, since this is the best performing sequence of the three normal LDSs discussed. In the first RLDS, $\underline{U} \sim U(0,1)^d$ is generated with the RNG, `Random` (R Development Core Team and contributors worldwide 2011) and then added to each point of the SS modular 1 (Sobol' R) (Glasserman 2004, pp. 320–321). The next three RLDSs 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) (Owen 1998, Tezuka and Faure 2003). These can all be found in the package `fOptions` (Wuertz 2010) in `R` (R Development Core Team 2009).

4.2 Measuring efficiency and results for QMC techniques

The results section will be divided into two parts. 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.

The first method uses the Root Mean Squared Error (RMSE) to compare the different values for the number of points used. After this the Relative RMSE (RRMSE) will be used to compare the different values for the dimension.

4.2.1 Constant dimensionality

The first results will be obtained with the equation

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

with m ERBPO problems with true values $\alpha_1, \ldots, \alpha_m$ and n-point approximations denoted by $\widehat{\alpha}_1(n), \ldots, \widehat{\alpha}_m(n)$. 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 with $\widehat{\alpha}_{MC,i}(N^*)$ with $N^* \to \infty$. The LLN implies that this will be arbitrarily close to the true value (given N^* is sufficiently large). The RMSE equation now changes to

$$\begin{aligned} RMSE(n) &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \widehat{\alpha}_{MC,i}(N^*))^2} \\ &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \alpha_i + \alpha_i - \widehat{\alpha}_{MC,i}(N^*))^2} \\ &= \sqrt{\underbrace{A}_{\to 0} + \underbrace{B}_{\to d_{m,N^*}} + \underbrace{\text{crossproduct}}_{\to 0}} \end{aligned}$$

where

$$\begin{aligned} A &= \frac{1}{m}\sum_{i=1}^{m}(\widehat{\alpha}_i(n) - \alpha_i)^2 \\ B &= \frac{1}{m}\sum_{i=1}^{m}(\alpha_i - \widehat{\alpha}_{MC,i}(N^*))^2. \end{aligned}$$

Thus, $RMSE(n) \to d_{m,N^*}$ as $n \to \infty$ and $RMSE(n) \to 0$ as $N^* \to \infty$. In reality, the RMSE will always converge to a number d_{m,N^*} due to the fact that $N^* \to \infty$ is computationally impossible. Nevertheless, the different methods may still be compared on how fast they converge to this value. Figure 3 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 $n \in \{4^4, 4^5, 4^6,$

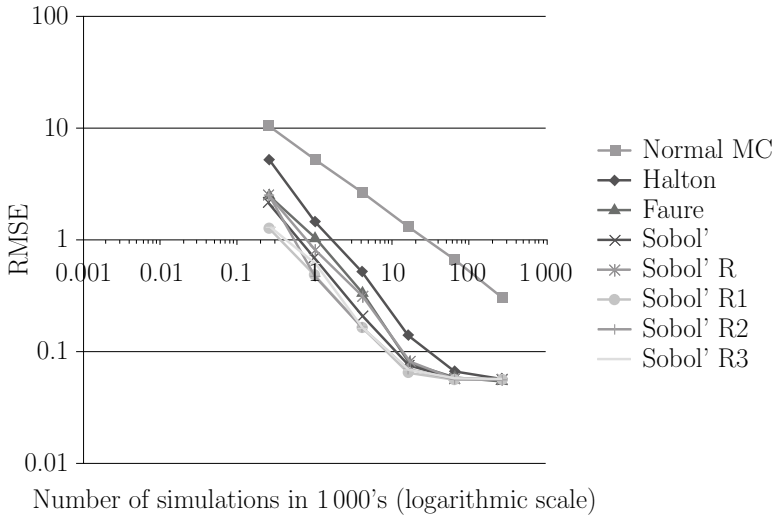


Figure 3: *RMSE for different RLDSs over increasing values of n.*

4^7, 4^8, 4^9}. Here, the most important input to the combinations is once again the number of simulations, n. The other parameters were simply chosen to group the simulations.

Figure 3 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 faster to the value d_{m,N^*}. This gives two advantages: faster convergence, and because this is a randomised sequence, the construction of a CI.

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

4.2.2 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. Glasserman (2004, p. 327) suggests using the RRMSE to compare the different sequences when considering increasing dimensionality. The formula is given by

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

with m ERBPO problems with true values $\alpha_1, \ldots, \alpha_m$ and n-point approximations denoted by $\widehat{\alpha}_1(n), \ldots, \widehat{\alpha}_m(n)$. 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 $N^* \to \infty$.

In the results given in Figure 4, n was chosen as 5 120 and N^* as 900 000. The time between rebalancing was carefully chosen such that the dimension of the problems changes from 20,

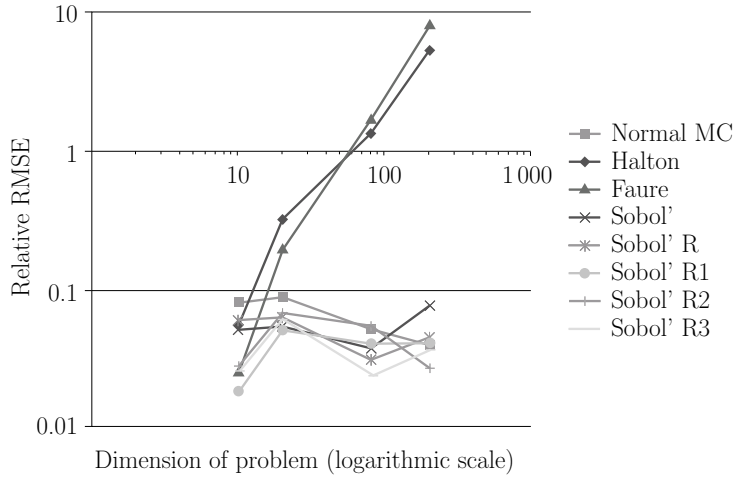


Figure 4: *RRMSE for different RLDSs over increasing sizes of the dimension.*

40, 80 up 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 were chosen to obtain the desired dimension of the problems. They are $\tau \in \{2,\ 1,\ 0.25,\ 0.1\}$. Here, the most important input to the combinations is the time between rebalancing (τ). The other parameters were simply chosen to group the simulations.

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). Comparing the other LDSs it is clear that the other methods all produce smaller RRMSEs than that of the normal MC method. However, the efficiency decreases as the dimension increases. For example, when $d = 10$ the Sobol' R1 method produced an RRMSE which was significantly smaller than the normal MC method, but when $d = 200$ they produced almost the same RRMSE.

5 Conclusion

Monte Carlo techniques may be used as a method to price a variety of different exotic options. This article aimed to find the best Monte Carlo technique to price the ERBO. A simplistic approach was refined using a mathematical proof after which different VRT were applied to help reduce the size of the error. Convergence was then increased with the help of different Quasi-Monte Carlo and Randomised Quasi-Monte Carlo techniques. The final combined algorithm for optimal pricing of the ERBO is given in the Appendix. This can be programmed in any mathematical/statistical package. It would be advised to program it in `R` (R Development Core Team 2009) as the Randomised Quasi-Monte Carlo techniques are readily available. Thus, by using the Refined Monte Carlo 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 are obtained.

Only the pricing of the ERBPO was discussed in this paper, the algorithm was adapted

to price the ERBCO. Note, however, that the price of the ERBCO can be found with the help of the put-call parity that can easiliy be derived using the normal arguments for the plain vanilla put-call parity. That is

$$c_t + Ke^{-r(T-t)} = p_t + \Pi_t + \Pi_t \sum_{j=1}^{a} \left(v_j \left(1 - e^{-q_j(T-t)} \right) \right),$$

with c_t and p_t the prices at time t of the ERBCO and ERBPO respectively.

Although the initial research question was answered, there still exist some open questions which can serve as future research topics. Further research could be performed on combining different VRT to possibly find an improvement on the classical VRT. Another possibility includes (a) determining whether different input parameters may be considered and (b) a method on how to predict which VRT would reduce the SE the most can be found. This could be done with Linear Discrimenant Analysis, Classification and Regression Trees or other Data-mining techniques on previous simulations.

Finally, this article showed that the Refined Monte Carlo method decreased the computational time of the value of the ERBO by approximately 98% (compared to the Simplified Monte Carlo method); the error of the estimates was smaller than it was for normal Monte Carlo approximately 95% of the time using different Variance Reduction Techniques; and by applying Quasi-Monte Carlo methods, the number of simulations needed to obtain the same accuracy than normal Monte Carlo decreased by approximately 99%. Hence, advanced simulation procedures are worthwhile to implement when pricing exotic type derivatives using Monte Carlo simulation.

References

[1] Black F & Scholes M, 1973, *The pricing of options and corporate liabilities*, Journal of Political Economy, **81(3)**, pp. 637–654.

[2] Boyle P, 1977, *Options: A Monte Carlo approach*, Journal of Financial Economics, **4**, pp. 323–338.

[3] Carnell R, 2009, *LHS 0.5 Edition*, [Online], [Cited April 25^{th} 2012], Available from: `http://cran.r-project.org/web/packages/lhs/index.html`.

[4] Chan NH & Wong HY, 2006, *Simulation Techniques in Financial Risk Management*, John Wiley & Sons, Inc., New Jersey (NJ).

[5] Faure H, 1982, *Discrépance des suites associées à un systéme de numèration*, Acta Arithmetica, **41**, pp. 337–351.

[6] Financial Services Authority, 2011, *Solvency II*, [Online], [Cited April 25^{th} 2012], Available from: `http://www.fsa.gov.uk/pages/About/What/International/solvency/index.shtml`.

[7] Financial Services Board, 2010, *Solvency Assessment and Management (SAM) Roadmap*, [Online], [Cited April 25^{th} 2012], Available from: `ftp://ftp.fsb.co.za/public/media/SAMROADMAP03112010.pdf`.

[8] Glasserman P, 2004, *Monte Carlo Methods in Financial Engineering*, Springer Science and Business Media, New York (NY).

[9] Halton JH, 1960, *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numerische Mathematic, **2**, pp. 84–90.

[10] Hammersley JM, 1960, *Monte Carlo methods for solving multivariable problems*, Annals of the New York Academy of Sciences, **86**, pp. 844–874.

[11] McKay MD, Conover WJ & Beckman RJ, 1979, *A comparison of three methods for selecting input variables in the analysis of output from a computer code*, Technometrics, **21**, pp. 239–245.

[12] Owen AB, 1998, *Scrambling Sobol' and Niederreiter-Xing points*, Journal of Complexity, **14(4)**, pp. 466–489.

[13] R Development Core Team, 2009, *R: A language and environment for statistical computing*, [Online], [Cited April 25^{th} 2012], Available from: `http://www.R-project.org`.

[14] R Development Core Team, 2011, *The R base package, 2.12.1 Edition*, [Online], [Cited April 25^{th} 2012], Available from: `http://www.R-project.org`.

[15] Rice JA, 2007, *Mathematical statistics and data analysis*, Thomson Higher Education, Belmont (CA).

[16] Sobol' IM, 1967, *On the distribution of points in a cube and the approximate evaluation of integrals*, USSR Journal of Computational Mathematics and Mathematical Physics, **7**, pp. 784–802.

[17] Stein M, 1987, Large sample properties of simulations using Latin hypercube sampling, Technometrics, **29**, pp. 143–151.

[18] Tezuka S & Faure H, 2003, *I-binomial scrambling of digital nets and sequences*, Journal of Complexity, **19**, pp. 744–757.

[19] Wuertz D, 2010, *fOptions*, [Online], [Cited April 25^{th} 2012], Available from: `http://cran.r-project.org/web/packages/fOptions/index.html`.

Appendix: Programmable algorithm to value the ERBO

The pricing algorithm, which may be implemented in any capable statistical sotftware program, for the ERBO becomes:

Algorithm 1: Price of the ERBO

Input : n, Π_0, τ, $\underline{v}$, T, r, K, $\underline{\sigma}$, Σ, *type*, $\underline{q}$.

Output : $\bar{O}_{CV}$ – the estimated price of the ERBO, SE – the SE of the estimate, CI – 95% CI.

1 `MAIN`(n, Π_0, τ, $\underline{v}$, T, r, K, $\underline{\sigma}$, Σ, *type*, $\underline{q}$);

2 $\bar{O}_{CV} \leftarrow \sum_{i=1}^{n_2} \underline{P}_{CV}[i]/n_2$;

3 $S_P \leftarrow \sqrt{\sum_{i=1}^{n_2} (\underline{P}_{CV}[i] - \bar{O}_{CV})^2/(n_2 - 1)}$;

4 $SE \leftarrow S_P/\sqrt{n_2}$;

5 $CI \leftarrow [\bar{O}_{CV} - 1.96 \cdot SE, \bar{O}_{CV} + 1.96 \cdot SE]$;

Algorithm 2: MAIN

Input : n, Π_0, τ, $\underline{v}$, T, r, K, $\underline{\sigma}$, Σ, *type*, $\underline{q}$.
Output : All scalars, vectors and matrices generated in this algorithm can be used in future computations.

if $type = call$ **then**
  $I \leftarrow -1$;
end
else if $type = put$ **then**
  $I \leftarrow 1$;
end
$n_1 \leftarrow \lfloor 0.05 \cdot n \rfloor$, $n_2 \leftarrow n - n_1$, $\underline{P} \leftarrow \underline{0}$ [vector of length n_1], $\underline{CV} \leftarrow \underline{0}$ [vector of length n_1];
if ($T \mod \tau = 0$) **then**
  $\underline{\Delta t} \leftarrow [\tau, \tau, \ldots, \tau]$ (length T/τ);
end
else
  $\underline{\Delta t} \leftarrow [\tau, \tau, \ldots, \tau, T \mod \tau]$ (length $\lceil T/\tau \rceil$);
end
Generate $A \leftarrow a \cdot \lceil T/\tau \rceil \times n_1$ matrix containing the SS;
$Z_j, \epsilon_j \leftarrow 0$ [$\lceil T/\tau \rceil \times n_1$ matrices for $j = 1, \ldots, a$];
for $i = 1$ *to* n_1 **do**
  $\Pi \leftarrow \Pi_0$, $CV_j \leftarrow \Pi_0$ for $j = 1, \ldots, a$;
  for $\ell = 1$ *to* $\lceil T/\tau \rceil$ **do**
    Calculate $Z_j[\ell, i]$ using the IPIT and transform with the Cholesky decomposition to obtain $\epsilon_j[\ell, i]$ with the i^{th} column of A, for $j = 1, \ldots, a$;
    $\Theta_j \leftarrow \exp\left(\left(r - q_j - \frac{\sigma_j^2}{2}\right)\underline{\Delta t}[\ell] + \sigma_j \epsilon_j[\ell, i]\sqrt{\underline{\Delta t}[\ell]}\right)$ for $j = 1, \ldots, a$;
    $\Pi \leftarrow \Pi \cdot \sum_{j=1}^{a} v_j \Theta_j$, $CV_j \leftarrow CV_j \cdot \Theta_j$ for $j = 1, \ldots, a$;
  end
  $\underline{P}[i] \leftarrow \max\{I \cdot (K - \Pi), 0\} e^{-rT}$, $\underline{CV}[i] \leftarrow \sum_{j=1}^{a} \left(\max\{I \cdot (K - CV_j), 0\} e^{-rT}\right)$;
end
$\bar{P} \leftarrow \sum_{i=1}^{n_1} \underline{P}[i]/n_1$, $S_P \leftarrow \sqrt{\sum_{i=1}^{n_1} (\underline{P}[i] - \bar{P})^2/(n_1 - 1)}$;
$\bar{CV} \leftarrow \sum_{j=1}^{a} BS_j(S_0 = \Pi_0, T = T, r = r, \sigma = \sigma_j, K = K, q = q_j, type = type)$;
$\sigma_{P,CV} \leftarrow \sum_{i=1}^{n_1} (\underline{P}[i] - \bar{P})(\underline{CV}[i] - \bar{CV})/n_1$, $c^* \leftarrow -\sigma_{P,CV}/S_P^2$;
$\underline{P} \leftarrow \underline{0}$ [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 $A \leftarrow a\lceil T/\tau \rceil \times n_2$ matrix containing the SS, $Z_j, \epsilon_j \leftarrow 0$ [$\lceil T/\tau \rceil \times n_2$ matrices for $j = 1, \ldots, a$];
for $i = 1$ *to* n_2 **do**
  $\Pi \leftarrow \Pi_0$, $CV_j \leftarrow \Pi_0$ for $j = 1, \ldots, a$;
  for $\ell = 1$ *to* $\lceil T/\tau \rceil$ **do**
    Calculate $Z_j[\ell, i]$ using the IPIT and transform with the Cholesky decomposition to obtain $\epsilon_j[\ell, i]$ with the i^{th} column of A, for $j = 1, \ldots, a$;
    $\Theta_j \leftarrow \exp\left(\left(r - q_j - \frac{\sigma_j^2}{2}\right)\underline{\Delta t}[\ell] + \sigma_j \epsilon_j[\ell, i]\sqrt{\underline{\Delta t}[\ell]}\right)$ for $j = 1, \ldots, a$;
    $\Pi \leftarrow \Pi \cdot \sum_{j=1}^{a} v_j \Theta_j$, $CV_j \leftarrow CV_j \cdot \Theta_j$ for $j = 1, \ldots, a$;
  end
  $\underline{\Pi}_T[i] \leftarrow \Pi$, $\underline{P}[i] \leftarrow \max\{I \cdot (K - \Pi, 0), 0\} e^{-rT}$, $\underline{CV}[i] \leftarrow \sum_{j=1}^{a} \left(\max\{I \cdot (K - CV_j), 0\} e^{-rT}\right)$,
  $\underline{P}_{CV}[i] \leftarrow \underline{P}[i] + c^*(\underline{CV}[i] - \bar{CV})$;
end