CHALMERS | GÖTEBORG UNIVERSITY

MASTER'S THESIS

Sensitivity Analysis of the Exotic Option Triple Obligation & Vega-Hedging in Bates' Market Model and the Associated Transaction Cost

SIMON ÖSTERBERG

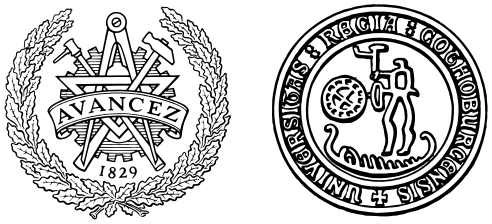
Department of Mathematical Statistics
CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG UNIVERSITY
Göteborg, Sweden 2005

Thesis for the Degree of Master of Science (20 credits)

Sensitivity Analysis of the Exotic Option Triple Obligation & Vega-Hedging in Bates' Market Model and the Associated Transaction Cost

Simon Österberg

CHALMERS | GÖTEBORG UNIVERSITY



Department of Mathematical Statistics
Chalmers University of Technology and Göteborg University
SE – 412 96 Göteborg, Sweden
Göteborg, June 2005

Thesis for the degree Master of Science in Engineering Physics
with specialization in Engineering Mathematics

Sensitivity Analysis of the Exotic Option Triple Obligation & Vega-Hedging in Bates' Market Model and the Associated Transaction Cost

Simon Österberg

Chalmers University of Technology
In collaboration with Svenska Handelsbanken AB

Göteborg, 2005

Abstract

This thesis deals with two aspects of mathematical finance; In the first part the Monte Carlo method is used for analyzing the exotic option Triple Obligation, by deriving the price and Greeks with corresponding confidence intervals. Estimations for the price, Delta, Vega, and the Correlation risk are successfully derived, while estimations for Gamma, Cross-Gamma, Vomma and Theta are insufficiently accurate due to the poor convergence rate provided by the Monte Carlo method. Further, a method for approximating Triple Obligation by a less complex option, for which there exist an analytical expression for the price, is presented. Using the results from the Monte Carlo simulations for calibration, we can attain valid estimations for the price, Delta, Vega and the Correlation risk within a fraction of the Monte Carlo computational time.

In the second part, Bates' market model is introduced to cope with the volatility smile phenomenon. By means of Fourier methods, analytical expressions for the price, Delta and Vega of call and digital options are derived and the Fast Fourier Transform (FFT) is presented and customized for the evaluation of these expressions. These results are applied to Vega-hedge a digital option and to calculate the associated transaction cost, originating from the call option price spread.

Acknowledgements

I would like to express my sincere thanks to both Mr. Tor Nordqvist and Prof. Patrik Albin. I am deeply grateful to Mr. Nordqvist, Head of Quantitative Analysis in the Equity Derivative Group at Svenska Handelsbanken Capital Markets, for providing me with such an interesting topic, and for the support and rewarding discussions along the way. I give my great appreciation to supervisor Prof. Patrik Albin, Department of Mathematical Statistics at Chalmers University of Technology, for insightful comments and suggestions. Finally, I would like to thank my family, my source of inspiration and motivation, without whom I never would have made it.

Contents

1 Introduction **1**

2 Sensitivity Analysis of the Exotic Option Triple Obligation **2**
2.1 Prerequisites . . . 2
2.1.1 Monte Carlo . . . 2
2.1.2 Quasi-Monte Carlo . . . 3
2.1.3 Pricing and the Greeks . . . 4
2.2 Description of Triple Obligation . . . 6
2.3 Generating sample paths . . . 6
2.4 Convergence results . . . 7
2.5 Approximation of Triple Asian-Digital . . . 8

3 Vega-Hedging in Bates' Market Model and the Associated Transaction Cost **10**
3.1 Prerequisites . . . 10
3.1.1 Bates' market model . . . 10
3.1.2 Pricing . . . 11
3.1.3 Fast Fourier transform . . . 12
3.2 Calculation of the Greeks . . . 13
3.2.1 Call option . . . 13
3.2.2 Digital option . . . 15
3.3 Calibration . . . 16
3.4 Hedging . . . 17
3.4.1 Generating sample paths . . . 18
3.4.2 Portfolio updating . . . 18
3.5 Transaction cost . . . 19

A Additional Plots **21**
A.1 Convergence results . . . 21
A.1.1 Price . . . 21
A.1.2 Delta . . . 22
A.1.3 Gamma . . . 23
A.1.4 Cross-Gamma . . . 24
A.1.5 Vega . . . 25
A.1.6 Vomma . . . 26
A.1.7 Theta . . . 27
A.1.8 Correlation risk . . . 28

B MATLAB Code **30**

Bibliography **51**

1 Introduction

This thesis deals with two aspects of mathematical finance; The first part concerns the pricing and the price derivatives with respect to market parameters, *the Greeks*, of the exotic option *Triple Obligation*, derived and analyzed by means of Monte Carlo (MC). These results are compared to results derived by means of Quasi-Monte Carlo (QMC). When pricing and hedging exotic options MC is a commonly used tool. Even though it provides an easy implementation and the possibility to evaluate a wide variety of financial instruments, there exists an considerable downside; The convergence of order $\mathcal{O}(n^{-1})$, where n is the number of iterations, calls for great computational effort and time to provide sufficiently accurate results.

To evade the computational effort required by MC one would like to find another option, similar to Triple Obligation, to which there exist an analytic expression for the price. Here, this option is chosen to be a *Triple Digital* and optimize the option parameters such that it mimics Triple Obligation in terms of the price and the Greeks.

In the second part of this thesis attention is attracted to the market model introduced by Bates, where stochastic volatility and price process jumps are introduced, in order to obtain a more accurate description of a market. When dealing with models that introduces stochastic volatility the methodology for hedging an option needs to be revised; In the Black-Scholes market model (BS), which is the most commonly used model, one can hedge all options by a risk-free Delta-hedge strategy, but now a risk-free strategy is also needed for coping with the price fluctuations caused by the stochastic volatility, a Vega-hedge strategy. By deriving analytical results for the price and the Greeks of a call option and a digital option under Bates' market model, such a strategy is derived and applied to the hedging of a digital option.

2 Sensitivity Analysis of the Exotic Option Triple Obligation

In this section an analysis is performed of the exotic option *Triple Obligation* by means of deriving the price and the derivatives of the price w.r.t. the market parameters, *the Greeks*. The problem is analyzed seen from an investment bankers point of view. Thus, one wants to attain sufficiently accurate results while keeping the computational effort to a minimum. To perform this reduction a less complex option, to which we have an analytical price, is optimized to approximate the original. This is done by minimizing the difference between the prices and the Greeks of the two options.

2.1 Prerequisites

2.1.1 Monte Carlo

The most common use for Monte Carlo (MC) methods in mathematical finance is for evaluating an expected value of a function of a random variable $f(X)$ with probability density function $\Psi(\mathbf{x})$ over $\mathbf{x} \in \mathbb{R}^n$, i.e.

$$v = \mathbf{E}[f(X)] = \int_{\mathbb{R}^n} f(\mathbf{x})\Psi(\mathbf{x})d\mathbf{x}.$$

Drawing N random numbers X_i from the distribution Ψ, arithmetic mean of f can be calculated, resulting in the Monte Carlo estimator

$$\hat{v} \equiv \frac{1}{N}\sum_{i=1}^{N} f(X_i)$$

of v. By defining

$$\sigma_f^2 = \int_{\mathbb{R}^n} (f(\mathbf{x}) - v)^2 \Psi(\mathbf{x})d\mathbf{x}$$

one realizes that, by virtue of the central limit theorem, $\hat{v}$ is approximately distributed as

$$\mathbf{N}\left(v, \frac{\sigma_f}{\sqrt{N}}\right). \tag{1}$$

The strength of MC methods reveals itself when the dimensionality of the domain increases; For a given d-dimensional hypercube with volume λ^d the error $\epsilon_{lattice}$ of approximating the integral of a given function by linear means decreases as $\mathcal{O}(\lambda^2)$. In a regular grid over a d-dimensional domain, the number of points N is proportional to λ as

$$N \propto \lambda^{-d},$$

implying

$$\lambda \propto N^{-1/d}.$$

Thus for a fixed number N of evaluations of the function, the relative error relates to N and d according to

$$\epsilon_{lattice}(N, d) \propto N^{-2/d},$$

to be compared with the relative error from MC

$$\epsilon_{MC}(N, d) \propto N^{-1/2}.$$

This aspect is highly valuable for application in mathematical finance, as the dimensionality of a problem is given by

$$d = kl,$$

where k represents the number of underlying assets and l represents the number of timesteps in the time discretization of the asset trajectories, tends to be large for path-dependent instruments.

2.1.2 Quasi-Monte Carlo

Quasi Monte Carlo (QMC) methods differ from ordinary MC methods in that they do not attempt to mimic randomness; By using low-discrepancy series, the aim is to generate points in space that are too evenly distributed to be random. There are many such series and in this thesis focus will be put on the sequence introduced by I.M. Sobol, further explained in [**Sob**]. It can be shown that QMC using these low-discrepancy methods under the right circumstances can attain a convergence rate of $\mathcal{O}(c(d)\ln(N)^d/N)$, where d is the problem dimension. This convergence is asymptotically much faster then for MC but the problem lies in the fact that the coefficient $c(d)$ may depend on the dimension and thus, for any one high-dimensional calculation, one cannot a priori know the convergence rate; Number-theoretical results can no longer explicitly determine if, nor how, QMC outperforms MC and one is left to study empirical results.

The problems occurring in high dimensionality can be visualized by, for example, studying two dimensions of the Sobol sequence as done in Figure 1. Some dimensions have an apparent correlation resulting in an uneven spread of points throughout the dimension, and this behavior becomes more common as the dimensionality increases. This aspect of QMC is often misinterpreted and as shown in [**Jäc**], by initiating the Sobol sequence in a correct manner, one can obtain at least the same convergence rate as for a ordinary MC, even in higher dimensions. The QMC method will not be further analyzed as it is not central in this thesis. Instead, results from QMC by the use of the Sobol sequence are compared to derived MC results.

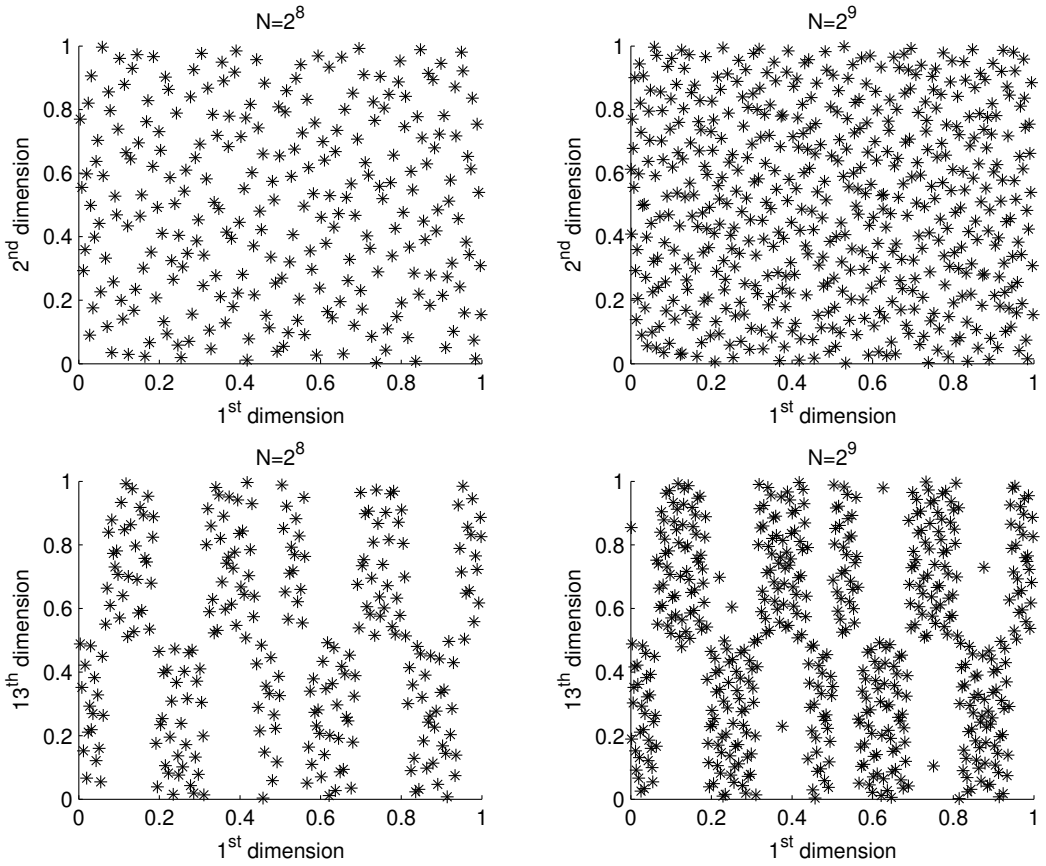


Figure 1: N first numbers in a 30-dimensional Sobol sequence

2.1.3 Pricing and the Greeks

In this section the BS market model [**BS**] is introduced, where d assets are modeled by geometric Brownian motions

$$dS_i(t) = rS_i(t)dt + \sigma_i S_i(t)dX_i(t), \quad i = 1, 2, ..., d, \tag{2}$$

under the risk-neutral measure $\mathbb{Q}$, where X_i and X_j have constant correlation ρ_{ij} and r is the constant risk-neutral interest rate. The price of a derivative can in this scenario be determined through use of the *risk-neutral pricing formula:*

$$\Pi(t) = \mathbb{E}_{\mathbb{Q}}[e^{-r(T-t)}g(T) \mid \mathfrak{F}_t],$$

where $\mathfrak{F}_t$ is the filtration of $X(t)$, g is the payoff-function for the derivative and T is the maturity time. But not only the price is of relevance when analyzing derivatives; The *Greeks*, i.e. the derivatives of the price w.r.t. various market parameters, play

Greek	Definition	Denotation
Delta	Π_{S_i}	Δ_i
Gamma	$\Pi_{S_iS_i}$	Γ_i
Cross-gamma	$\Pi_{S_iS_j}$	Γ_{ij}
Theta	Π_t	Θ
Vega	Π_{σ_i}	ζ_i
Vomma	$\Pi_{\sigma_i\sigma_i}$	η_i
Correlation risk	Π_ρ	υ

Table 1: The Greeks

an important role e.g. when hedging. The definitions and denotations are stated in Table 1.

When calculating the first order Greeks by means of MC or QMC, one often use the *first order finite difference approximation*

$$\widehat{\Pi}_\xi(t) = \frac{\Pi(t,\xi+d\xi)-\Pi(t,\xi)}{d\xi} = \Pi_\xi(t) + \mathcal{O}(d\xi). \tag{3}$$

Assuming a sufficiently large number of iterations n the result in (1) can be used to derive a $(1-\alpha)\%$ confidence interval for the calculated Greek as

$$\mathbb{P}\left\{ e^{-r(T-t)}\widehat{\Pi}_\xi(t) \in \frac{1}{d\xi}\overline{Y} \pm \lambda_{\alpha/2}\frac{s_Y}{\sqrt{n}d\xi}\middle|\mathfrak{F}_t\right\} = 1-\alpha,$$

where $Y = g(T,\xi+d\xi)-g(T,\xi)$, $\lambda_{\alpha/2} = \Phi^{-1}(\alpha/2)$ if Φ is the cumulative normal distribution function and s_Y is the sample standard deviation of Y,

$$s_Y^2 = \overline{Y^2} - \overline{Y}^2.$$

Regarding the second order Greeks, the corresponding results for the *second order finite difference approximations* are

$$\widehat{\Pi}_{\xi\xi}(t) = \frac{\Pi(t,\xi+d\xi)-2\Pi(t,\xi)+\Pi(t,\xi-d\xi)}{d\xi^2} = \Pi_{\xi\xi}(t) + \mathcal{O}(d\xi^2),$$

and

$$\mathbb{P}\left\{ e^{-r(T-t)}\widehat{\Pi}_{\xi\xi}(t) \in \frac{1}{d\xi^2}\overline{Z} \pm \lambda_{\alpha/2}\frac{s_Z}{\sqrt{n}d\xi^2}\middle|\mathfrak{F}_t\right\} = 1-\alpha,$$

respectively, where $Z = g(T,\xi+d\xi)-2g(T,\xi)+g(T,\xi-d\xi)$. One can see that the relative error of the approximation, in both cases, depend on n as well as on $d\xi$, implying that for each fix value of n one has to determine the perturbation size $d\xi$ minimizing the total error.

2.2 Description of Triple Obligation

The derivative *Triple Obligation* is constructed to give an alternative to traditional interest investments by offering a larger payoff. The payoff is dependent on the evolution of three assets; If the assets all exceeds (or equals) their initial value at two measuring points, a yearly coupon of 6% is received. The maturity time is two years and the asset value at the measuring point is determined to be the mean asset value for the last month of the year, disregarding the 10 lowest values. An example is given in Table 2.

Asset	Ericsson B	H&M B	TeliaSonera	Coupon
Initial value	21.3	210.5	39.3	
Value after 1^{st} year	23.0	212.2	42.4	
Value equals or exceeds?	Yes	Yes	Yes	6.00%
Value after 2^{nd} year	18.6	225.9	36.1	
Value equals or exceeds?	No	Yes	No	0.00%
Total				106.00%

Table 2: Example of payoff for derivative Triple Obligation

Another derivative, *Triple Asian-Digital*, is now constructed: The maturity time is set to one year, and the derivative pays 1 if the three assets all exceeds (or equals) their initial value at the measuring point[1], or else pays 0. One can see that Triple Obligation is equivalent to a portfolio consisting of Triple Asian-Digital options and obligations, in the right proportions. Knowing this, one can now perform an analysis of the less complex derivative Triple Asian-Digital, from which results for Triple Obligation easily can be derived.

2.3 Generating sample paths

Recalling the BS model introduced in section 2.1.3 and defining the $d \times d$-dimensional correlation matrix

$$\Sigma_{ij} = \sigma_i \sigma_j \rho_{ij},$$

the notation

$$(\sigma_1 X_1(t), ..., \sigma_d X_d(t)) = BM(0, \Sigma)$$

is introduced. Now $BM(0, \Sigma)$ can be written as $A \cdot BM(0, I)$, where A is the Cholesky factorization of Σ, i.e. $AA^T = \Sigma$. Further, one can now write (2) as

$$\frac{dS_i(t)}{S_i(t)} = rdt + \sum_{j=1}^{d} A_{ij} W_j(t), \quad i = 1, ..., d,$$

[1]The measuring point is defined in analogous manner as for Triple Obligation

where $W(t) = (W_1(t), ..., W_d(t))$ is a d-dimensional Brownian motion. Sample paths can now be generated as

$$S_i(t_n) = S_i(t_{n-1}) \exp\left(\left(r - \frac{\sigma_i^2}{2}\right)(t_n - t_{n-1}) + \sqrt{t_n - t_{n-1}} \sum_{j=1}^{d} A_{ij} Z_j\right),$$

with the partition $0 = t_0 \leq ... \leq t_{n-1} \leq t_n \leq ... \leq t_N = T$ and Z_j are normally distributed random variables, $j = 1, ..., d$.

2.4 Convergence results

Calculations are now performed for the derivation of the price and the Greeks as functions of the number of iterations. The parameter setting are chosen as

$$\begin{aligned} S_1(0) &= 100, & S_2(0) &= 100, & S_3(0) &= 100, \\ \sigma_1 &= 0.20, & \sigma_2 &= 0.25, & \sigma_3 &= 0.30, \\ \rho_{12} &= 0.30, & \rho_{13} &= 0.40, & \rho_{23} &= 0.50, \\ t_i &= i/250, & i &= 0, ..., 250, \\ r &= 0.03, \end{aligned}$$

to represent a typical case of a stock market. Hereafter, when using the term relative error, values from QMC using the Sobol sequence and 2^{23} iterations are used as "exact" values, an assumption validated by arguments in Section 2.1.2. When plotting, results are displayed in two figures; The first figure shows the interval of the MC sample value within one standard deviation. The second figure describes the *Maximum absolute relative error* (MARE) of the above defined interval. In addition, results for the QMC metod are included. Further, these results are derived using three different perturbation settings:

$$\begin{aligned} \text{Large perturbations:} \quad & dS = dT = 100\, d\sigma_i = 100\, d\rho_{ij} = 10. \\ \text{Medium perturbations:} \quad & dS = dT = 100\, d\sigma_i = 100\, d\rho_{ij} = 1. \\ \text{Small perturbations:} \quad & dS = dT = 100\, d\sigma_i = 100\, d\rho_{ij} = 0.1. \end{aligned}$$

Plots are found in Appendix A.1.

Interpreting the results one can see that the calculations of the price have a relatively fast convergence, where the MARE is less than 1% when the number of iterations exceeds 10^5. When analyzing the calculations of the Delta the two sides of the perturbation size problem can bee seen; For large perturbations a faster convergence is attained but a obvious bias exists. The other scenario is for the small perturbations, with a slower convergence but a neglectable bias. Nevertheless, with

an appropriate perturbation size, results where the MARE is less than 5% can be obtained. In the case of the Gamma, the MARE indicates a poor approximation. But observing that the QMC result is oscillating around zero one can conclude that the approximation still can be useful. For the rest of the Greeks there are some general results; First, the slow convergence of MC is an obvious problem, especially when dealing with second order finite difference approximations. Another problem is the relatively large error, to a large extent depending on the small values of the second order Greeks. To summarize, the "naive" MC method is an adequate tool for performing e.g. a simpler Delta-hedge, but lacks accuracy when further trying to characterize the derivative Triple Asian-Digital. This gives a reason for finding an appropriate approximation.

2.5 Approximation of Triple Asian-Digital

To further simplify and accelerate the evaluation of Triple Asian-Digital, one wants to make an approximation by a less complex derivative. Thus, a derivative that resembles a Triple Asian-Digital, the *Triple Digital*, is to be analyzed. The payoff for this derivative is 1 if the three asset values exceeds (or equals) $K_i = KS_i(0)$, respectively, at maturity time T. This derivative can be priced analytically [**Nor**]; First, the Cholesky factorization of the correlation matrix is calculated:

$$AA^T = \begin{pmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{pmatrix} \quad \Longrightarrow \quad A = \begin{pmatrix} 1 & 0 & 0 \\ \rho_{12} & \sqrt{1-\rho_{12}^2} = d_{22} & 0 \\ \rho_{13} & \dfrac{\rho_{23}-\rho_{12}\rho_{13}}{d_{22}} = d_{32} & \sqrt{1-\rho_{13}^2-d_{32}^2} \end{pmatrix}.$$

Now, the price can be calculated as

$$\Pi_D = \frac{e^{-r\tau}}{\sqrt{2\pi}} \int_{p_{1x}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-p_1^2/2} \left\{ \int_{p_2 x}^{\infty} e^{-p_2^2/2} N(-p_3) dp_2 \right\} dp_1,$$

where

$$p_{1x} = -\frac{\ln(S_1/K_1) + h_1}{g_1}, \quad g_1 = \sigma_1\sqrt{\tau}, \qquad h_1 = (r - q_1 - \frac{\sigma_1^2}{2})\tau,$$

$$p_{2x} = -\frac{\ln(S_2/K_2) + h_2}{g_2}, \quad g_2 = \sigma_2\sqrt{\tau}d_{22}, \quad h_2 = (r - q_2 - \frac{\sigma_2^2}{2})\tau + \sigma_2\sqrt{\tau}d_{21}p_1,$$

$$p_{3x} = -\frac{\ln(S_3/K_3) + h_3}{g_3}, \quad g_3 = \sigma_3\sqrt{\tau}d_{33}, \quad h_3 = (r - q_3 - \frac{\sigma_3^2}{2})\tau + \sigma_3\sqrt{\tau}(d_{31}p_1 + d_{32}p_2).$$

The above expression is now evaluated by Simpson´s rule and, using the finite difference method as in (3) with the same perturbations used in the MC estimation,

the equivalent estimation for the price and Greeks of Triple Digital can now be calculated. The parameters that should be optimized for best fit are the maturity time T and the strike coefficient K. The calibration is performed by taking use of the results attained in Section 2.4, where values and standard deviation for the price and the Greeks where derived; The goal is for the approximation to derive values that lie within one standard deviation of the MC sample result. Thus, we define a penalty function V to be minimized, denoting values originating from Triple Asian-Digital by index DA and values originating from Triple Digital by index D:

$$\begin{aligned} & V(K,T) \\ &= \left(\frac{\Pi_D - \Pi_{DA}}{\sigma_{\Pi_{DA}}}\right)^2 + \left(\frac{\upsilon_D - \upsilon_{DA}}{\sigma_{\upsilon_{DA}}}\right)^2 + \sum_{i=1}^{3}\left(\frac{\Delta_{D,i} - \Delta_{DA,i}}{\sigma_{\Delta_{DA,i}}}\right)^2 + \sum_{i=1}^{3}\left(\frac{\zeta_{D,i} - \zeta_{DA,i}}{\sigma_{\zeta_{DA,i}}}\right)^2. \end{aligned}$$

This function is optimized w.r.t. the result from the MC calculation in Section 2.4 with 2^{20} iterations, here by MATLAB routine `lsqnonlin`, producing the optimal parameters

$$\begin{aligned} T &= 0.60596, \\ K &= 0.97413, \end{aligned}$$

and the price and Greeks according to

$$\begin{aligned} \Pi_D &= 0.24856, \\ \Delta_{D,1} &= 0.00637, \\ \Delta_{D,2} &= 0.00528, \\ \Delta_{D,3} &= 0.00443, \\ \zeta_{D,1} &= -0.15942, \\ \zeta_{D,2} &= -0.12515, \\ \zeta_{D,3} &= -0.10271, \\ \upsilon_D &= 0.08859. \end{aligned}$$

When analyzing the results from the optimization one can see that all terms lie within one standard deviation from the target value; The approximation produces an result with the same accuracy as a MC calculation with 2^{20} iterations, by a triple integral evaluation. In terms of computational time this approximation is a great improvement and is always preferred, when one only needs to derive the above specified Greeks.

3 Vega-Hedging in Bates' Market Model and the Associated Transaction Cost

When observing a stock price process on the market, one can choose to calculate the volatility in different manners. One way to look at it is as the variance of the stock price under a certain time interval, empirically derivable. The other way is to derive it theoretically from the market option prices. As the volatility is the only unknown parameter, it can be solved for from the equation

$$\Pi_{\text{BS}}(S, t, \sigma_{\text{implied}}(K, T); K, T) = \Pi_{\text{market}}(S, t; K, T),$$

where Π_{BS} is the BS price, Π_{market} is the market price and σ_{implied} is the sought *Black-Scholes implied volatility*. Using a call option in the above calculation we can, because of the monotonicity of the call price w.r.t. volatility, easily solve the equation using, for example, the Newton-Raphson method.

In practice, all markets exhibits the volatility smile phenomenon; The volatility surface, consisting of the implied volatility as a function of the strike K and maturity T is non-constant with a characteristic behavior, elaborated in [**AHNW**]. Remembering that the BS model uses a constant volatility, one realizes that the BS model does not reveal the whole truth and a new, more sophisticated model is needed. There exists many market models trying to incorporate this volatility smile but no model has been generally accepted as every model has its advantages. In this section focus is put on the model introduced by Bates, where asset price jumps and stochastic volatility are introduced, and especially the consequences that arises; For an instrument in the BS market model there exist a risk-free delta-hedging strategy, but in Bates' market model a strategy coping with the price fluctuations caused by the stochastic volatility, a Vega-hedge strategy, is needed. By deriving analytical results for the price and the price derivatives of a call option and a digital option under Bates' market model, such a strategy is derived and applied to the hedging of a digital option.

3.1 Prerequisites

3.1.1 Bates' market model

A more general market model then in Section 2 is now introduced, consisting of an asset price process jumps and stochastic volatility;

$$dS(t) = \mu(S, t)dt + \sigma(S, t)dW(t) + JdN(t),$$

where $W(t)$ is a Brownian motion, μ and σ are adapted processes, $N(t)$ is a Poisson process with intensity λ and the random variable J as the jump size. Under the

risk-neutral measure $\mathbb{Q}$, the price process is described by the stochastic differential equation

$$dS(t) = (r - d - \lambda m)S(t)dt + \sigma(t)S(t)dW(t) + (e^J - 1)S(t)dN(t).$$

If now the processes are specified as proposed by Bates [**Bat**], the 1-dimensional market model is described by

$$\begin{aligned}
dS(t) &= (r - d - \lambda m)S(t)dt + \sqrt{V(t)}S(t)dW^s(t) + (e^J - 1)S(t)dN(t),\\
dV(t) &= \kappa(\Theta - V(t))dt + \epsilon\sqrt{V(t)}dW^v(t),\ V(0) = V,\\
\rho &= \mathbf{Cov}(dW^s(t), dW^v(t)),\\
\mathbb{P}(dN(t) = 1) &= \lambda dt,\\
\ln(J) &\sim \mathbf{N}(\nu, \delta),\\
m &= e^{\nu + 1/(2\delta^2)} - 1,
\end{aligned}$$

where $W^s(t)$ and $W^d(t)$ are Brownian motions. As a note, the process used to model the variance V is a CIR model, introduced by Cox-Ingersoll-Ross [**CIR**], a fact that will become useful when simulating trajectories.

3.1.2 Pricing

Now, observe an European derivative f, based on an Bates' market model asset, with maturity time T and payoff g_f. Further, the Fourier transform of g_f is denoted by $\hat{g}_f(z)$ and the domain where $\hat{g}_f(z)$ exists is denoted by *the strip of regularity* S_f. Making the substitution $x(T) = \ln S(T)$ the *characteristic function* of $x(T)$ can be written as

$$\phi_T(z) = \mathbb{E}[e^{izx}] = \int_{-\infty}^{\infty} e^{izx}\varpi_T(x)dx, \tag{4}$$

where ϖ_T is the risk-neutral density of $x(T)$. The price $\Pi_f(x(t))$ can now be derived by making use of *The Characteristic Formula* as stated in [**Sep**]:

Theorem 3.1.1 *Assume that $x(T)$ has the analytic characteristic function $\psi_T(z)$ with the strip of regularity $S_z = \{z : \alpha < \Im(z) < \beta\}$. Next, assume that $e^{z_i x}g_f(x) \in L^1(\mathbb{R})$ where z_i is located in the payoff strip S_f with transform $\hat{g}_f(z)$, $\Im(z) \in S_f$. Then, if $S_F = S_f \cap S_z$ is not empty, the option value is given by*

$$\Pi_f(x(t)) = \frac{e^{-r(T-t)}}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} \phi_T(-z)\hat{g}_f(z)dz,$$

where $z_i = \mathfrak{z}$, $z \in S_F = S_f \cap S_z$.

Proof: Using risk neutral pricing,

$$\begin{aligned}\Pi_f(x(t)) &= \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}g_f(x)\right] \\ &= e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}\left[\frac{1}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} e^{-izx(T)}\hat{g}_f(z)dz\right] \\ &= \frac{e^{-r(T-t)}}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} \mathbb{E}^{\mathbb{Q}}\left[e^{-izx(T)}\right]\hat{g}_f(z)dz \\ &= \frac{e^{-r(T-t)}}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} \phi_T(-z)\hat{g}_f(z)dz.\end{aligned}$$

□

3.1.3 Fast Fourier transform

The *Fast Fourier Transform* (FFT) is used for the computation of the sum

$$f(x) = \sum_{z=1}^{N} e^{-i(z-1)(x-1)2\pi/N}\hat{f}(z), \quad x = 1, 2, ..., N.$$

Choosing $N = 2^m$, $m \in \mathbb{N}$, the number of multiplications decrease from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\ln_2(N))$. If one wants to calculate the function F defined by

$$F(x) = \int_0^{\infty} e^{-izx}\Psi_T(z)dz,$$

the Trapezoid rule can be used, and by setting $z_j = \eta(j-1)$, the approximation

$$F(x) \approx \sum_{j=1}^{N} e^{-iz_j x}\Psi_T(z_j)\eta \tag{5}$$

is attained. The effective upper limit is $\alpha = N\eta$ and spacing size ω for return values is chosen to be

$$x_k = -\beta + \omega(k-1), \quad k = 1, 2, ..., N,$$

ranging from $-\beta$ to β, $\beta = N\omega/2$. Taking into calculation the restriction

$$\omega\eta = \frac{2\pi}{N}, \tag{6}$$

and improving the Trapezoid rule by Simpson´s rule weightings, an approximation, ready for FFT implementation, is attained:

$$F(x_k) = \sum_{j=1}^{N} e^{-i(j-1)(k-1)2\pi/N}e^{ibz_j}\Psi(z_j)\frac{\eta}{3}(3 + (-1)^j - \delta_{j-1}),$$

where δ is the Kronecker delta function; $\delta_n = 1$ for n=0, else 0. When choosing N, ω and η, there are a few aspects that needs to be considered:

- α should be sufficiently large for the approximation (5) to be valid.
- β should cover an relevant domain for x.
- (6) implies that choosing a finer grid for the integration makes the grid for x_k more sparse.

3.2 Calculation of the Greeks

Now make the substitutions $x(t) = \ln(S(t))$, $\tau = T - t$ and $X = x - \ln(K) + (r - d)\tau$. To be able to use the Theorem 3.1.1, an explicit expression for the characteristic function, defined in (4) is needed. This is done in [**Sep**] by first defining $\phi_T(z) = G(iz)$ and

$$Q(k, x, V, \lambda, \tau) = e^{-(-ik+1/2)\ln(K)} G(-ik + 1/2, x, V, \lambda, \tau),$$

then deriving the result from the expressions

$$\begin{aligned}
Q(k, x, V, \lambda, \tau) &= \exp\left((-ik + 1/2)X + A(k, \tau) + B(k, \tau)V + D(k, \tau)\lambda\right), \\
u &= \kappa - \rho\epsilon/2, \\
\xi &= \sqrt{k^2\epsilon^2(1 - \rho^2) + 2ik\rho\epsilon u + u^2 + \epsilon^2/4}, \\
\psi_\pm &= \mp(u + ik\rho\epsilon) + \xi, \\
A(k, \tau) &= -\frac{\kappa\Theta}{\epsilon^2}\left[\psi_+\tau + 2\ln\left(\frac{\psi_- + \psi_+ e^{-\xi\tau}}{2\xi}\right)\right], \\
B(k, \tau) &= -(k^2 + 1/4)\frac{1 - e^{-\xi\tau}}{\psi_- + \psi_+ e^{-\xi\tau}}, \\
D(k, \tau) &= \tau\left(e^{-ik(\nu+\delta^2/2)-(k^2-1/4)\delta^2/2+\nu/2} - 1 - (-ik + 1/2)(e^{\nu+\delta^2/2} - 1)\right).
\end{aligned}$$

3.2.1 Call option

Now the analytical expressions for the price, the Delta and the Vega of the European call option, with strike K and maturity time T, are to be derived. Having a payoff function

$$g_c(x) = \max(e^x - K, 0),$$

the price can by risk-neutral pricing, be written as

$$\begin{aligned}\Pi_c(x(t),t) &= \mathbb{E}^{\mathbb{Q}}[e^{-r(T-t)}g_c(x(T))] \\ &= e^{-r(T-t)}\left[\mathbb{E}^{\mathbb{Q}}[e^{x(T)}] + \mathbb{E}^{\mathbb{Q}}[\min(e^{x(T)},K)]\right] \\ &= e^{x(t)-(T-t)d} - \Pi_f(x,V,\lambda,t). \end{aligned} \tag{7}$$

Thus, the analysis is now focused on the option f with payoff function $g_f(x) = \min(e^x, K)$. The payoff Fourier transform is

$$\begin{aligned}\hat{g}_f(z) &= \int_{-\infty}^{\infty} e^{izx}\min(e^x,K)dx \\ &= \lim_{U\to\infty}\int_{-U}^{\ln K} e^{izx}e^x dx + K\lim_{U\to\infty}\int_{\ln K}^{U} e^{izx}dx \\ &= \lim_{U\to\infty}\frac{e^{(iz+1)x}}{iz+1}\Bigg|_{x=-U}^{x=\ln K} + K\lim_{U\to\infty}\frac{e^{izx}}{iz}\Bigg|_{x=\ln K}^{x=U} \\ &= \frac{e^{(iz+1)\ln K}}{iz+1} - 0 + K\left(0 - \frac{e^{iz\ln K}}{iz}\right) \\ &= \frac{K^{iz+1}}{iz+1} - \frac{K^{iz+1}}{iz} \\ &= \frac{K^{iz+1}}{z^2-iz}, \end{aligned}$$

with the strip of regularity $S_f = \{0 < \Im z < 1\}$. The price is now attained by applying Theorem 3.1.1

$$\Pi_f(x,V,\lambda,\tau) = \frac{K}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} e^{iz\ln(K)}\frac{G(-iz,x,V,\lambda,\tau)}{z^2-iz}dz.$$

Choosing $z_i = 1/2$ and making the substitution $k = z - i/2$ gives

$$\Pi_f(x,V,\lambda,\tau) = \frac{K}{2\pi}\int_{-\infty}^{\infty}\frac{Q(k,x,V,\lambda,\tau)}{k^2+1/4}dk.$$

Making yet another substitution

$$Q^*(k,V,\lambda,\tau) = e^{-(-ik+1/2)X}Q(k,x,V,\lambda,\tau),$$

the price is expressed as a Fourier integral

$$\Pi_f(x,V,\lambda,\tau) = \frac{Ke^{X/2}}{2\pi}\int_{-\infty}^{\infty} e^{-ikx}e^{-ik(\tau(r-d)-\ln(K))}\frac{Q^*(k,V,\lambda,\tau)}{k^2+1/4}dk. \tag{8}$$

The integral can be shown to be uniformly convergent, letting us change the order of integration and derivation yielding

$$\frac{\partial}{\partial x}\Pi_f(x,V,\lambda,\tau) = \frac{Ke^{X/2}}{2\pi}\int_{-\infty}^{\infty} e^{-ikx}e^{-ik(\tau(r-d)-\ln(K))}\frac{Q^*(k,V,\lambda,\tau)}{1/2+ik}dk,$$

and

$$\frac{\partial}{\partial V}\Pi_f(x,V,\lambda,\tau) = \frac{Ke^{X/2}}{2\pi}\int_{-\infty}^{\infty} e^{-ikx}e^{-ik(\tau(r-d)-\ln(K))}\frac{B(k,\tau)Q^*(k,V,\lambda,\tau)}{k^2+1/4}dk.$$

To summarize, the price of a call option under Bates' model is given in (7), the Delta is given by

$$\Delta_c(x(t),t) = e^{-(T-t)d} - e^{-x}\frac{\partial}{\partial x}\Pi_f(x,V,\lambda,t),$$

and the Vega is given by

$$\zeta_c(x(t),t) = -2\sqrt{V}\frac{\partial}{\partial V}\Pi_f(x,V,\lambda,t).$$

3.2.2 Digital option

An analogous analysis is now performed for the digital option with strike K and maturity time T: The payoff function

$$g_d(x,t) = \begin{cases} 1 & \text{if } e^x > K \\ 0 & \text{otherwise} \end{cases},$$

has the Fourier transform

$$\hat{g}_d(z) = \int_{-\infty}^{\infty} e^{izx}g_d(x,t)dx = \lim_{U\to\infty}\int_{\ln K}^{U} e^{izx}dx = \lim_{U\to\infty}\frac{e^{izU}}{iz} - \frac{K^{iz}}{iz} = -\frac{K^{iz}}{iz},$$

with the strip of regularity $S_d = \{\Im(z) < 0\}$. By yet again making use of Theorem 3.1.1 the price can be calculated as

$$\Pi_d(x,V,\lambda,\tau) = -\frac{1}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} e^{iz\ln(K)}\frac{G(-iz,x,V,\lambda,\tau)}{iz}dz.$$

Again, choosing $z_i = 1/2$ and making the substitution $k = z - i/2$ gives

$$\Pi_d(x,V,\lambda,\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{Q(k,x,V,\lambda,\tau)}{1/2-ik}dk.$$

The price can now be expressed as a Fourier integral

$$\Pi_d(x, V, \lambda, \tau) = \frac{e^{X/2}}{2\pi} \int_{-\infty}^{\infty} e^{-ikx} e^{-ik(\tau(r-d)-\ln(K))} \frac{Q^*(k, V, \lambda, \tau)}{1/2 - ik} dk. \tag{9}$$

It can be shown that this integral is uniformly convergent, letting us interchange integration and derivation, yielding

$$\frac{\partial}{\partial x}\Pi_d(x, V, \lambda, \tau) = \frac{e^{X/2}}{2\pi} \int_{-\infty}^{\infty} e^{-ikx} e^{-ik(\tau(r-d)-\ln(K))} Q^*(k, V, \lambda, \tau) dk,$$

and

$$\frac{\partial}{\partial V}\Pi_d(x, V, \lambda, \tau) = \frac{e^{X/2}}{2\pi} \int_{-\infty}^{\infty} e^{-ikx} e^{-ik(\tau(r-d)-\ln(K))} \frac{B(k, \tau) Q^*(k, V, \lambda, \tau)}{1/2 - ik} dk.$$

To summarize, the price is given in (9), the Delta is given by

$$\Delta_d(x(t), t) = e^{-x} \frac{\partial}{\partial x}\Pi_d(x, V, \lambda, \tau),$$

and the Vega is given by

$$\zeta_d(x(t), t) = 2\sqrt{V} \frac{\partial}{\partial V}\Pi_d(x, V, \lambda, \tau).$$

3.3 Calibration

Before implementing the model the parameters needs to be calibrated. By this is meant optimizing the model such that the BS implied volatility from Bates' call option prices, in the best possible way, resembles the BS implied volatility surface from the call option prices in the market we calibrate against. Further, the calibration criteria that are used are

- The at-the-money implied BS volatility is $\sigma_{\text{BS}} = 0.1325$. item If the Strike goes up by 10% the implied BS volatility goes down by 2%, i.e:

$$K \nearrow 10\% \implies \sqrt{V} = \sigma \searrow 2\%.$$

By using an optimization routine, here the MATLAB function `fminsearch`, to solve for the parameters $V, \kappa, \Theta, \epsilon, \rho, \lambda, \nu$ and δ, defined in Section 3.1.1, the least-square optimal solution is given by the optimal parameter setting

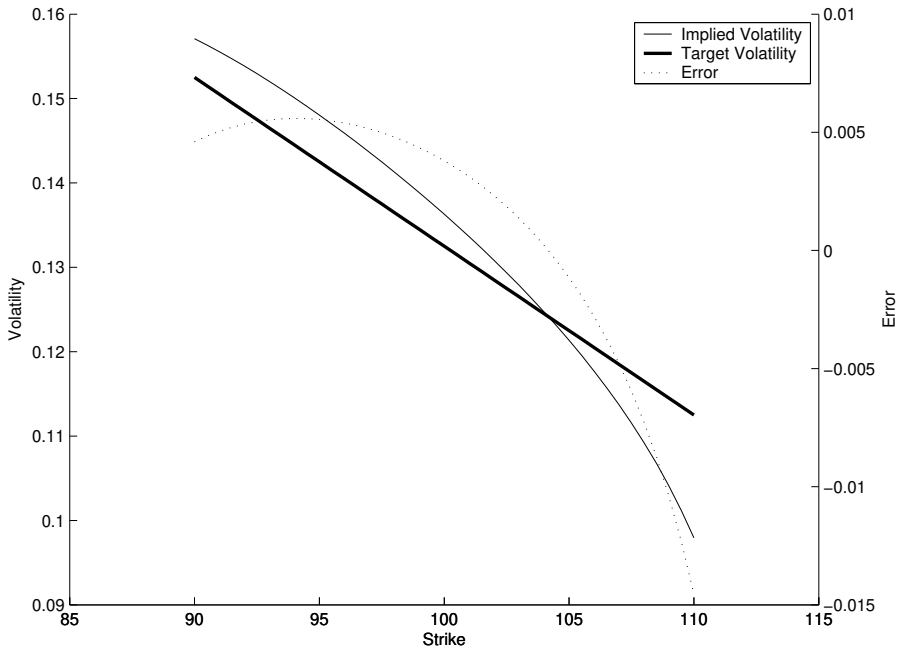


Figure 2: Calibration results for Bates' market model

$$
\begin{aligned}
V &= 0.0182, \\
\kappa &= 10.0077, \\
\Theta &= 0.0149, \\
\epsilon &= 0.4731, \\
\rho &= -0.4921, \\
\lambda &= 0.3398, \\
\nu &= -0.2090, \\
\delta &= 0.2553.
\end{aligned}
$$

3.4 Hedging

Having a calibrated model and analytical expressions for the relevant Greeks, ready to be implemented by FFT, all the means for hedging a digital option is at our disposal. The approach is; for each timestep t_i,

1. Simulate the value $V(t_{i+1})$.

2. Simulate the correlated value $S(t_{i+1})$.

3. Update the hedge portfolio by trading call options, making the portfolio Vega equal to the digital option Vega.

4. Update the hedge portfolio by trading stocks, making the portfolio Delta equal to the digital option Delta.

3.4.1 Generating sample paths

The transition density for the CIR model, that are used to model the variance V, is in [**Gla**] shown to be

$$V(t) = \frac{\epsilon^2(1-e^{-\kappa(t-u)})}{4\kappa}\chi^2_{\text{df}}\left(\frac{4\kappa e^{-\kappa(t-u)}}{\epsilon^2(1-e^{-\kappa(t-u)})}V(u)\right), \quad t > u,$$

where

$$\text{df} = \frac{4\Theta\kappa}{\epsilon^2},$$

and $\chi^2_\nu(\mu)$ is a noncentral chi-square distributed random variable with ν degrees of freedom and noncentrality parameter μ. By a simple Euler discretization the attained results can be transformed to terms of a Brownian motion increment;

$$W^v(t_{i+1}) - W^v(t_i) = \frac{V(t_{i+1}) - V(t_i) - \kappa(\Theta - V(t_i))(t_{i+1}-t_i)}{\epsilon\sqrt{V(t_i)}}.$$

One can now derive the value for the correlated Brownian increment as

$$W^s(t_{i+1}) - W^s(t_i) = \rho(W^v(t_{i+1}) - W^v(t_i)) + \sqrt{(1-\rho^2)(t_{i+1}-t_i)}Z_{i+1},$$

where Z_i are normally distributed random variables, which in turn makes it possible to simulate the evolution of the asset S by following the principles in Section 2.3.

3.4.2 Portfolio updating

Next, the number of call options and stocks that are needed to get a Greek-neutral portfolio needs to be calculated. A few simplifications are made to make the problem more manageable:

- The portfolio is hedged by trading new call options each day, i.e. we keep all call options that are traded in previous days.

- The maturity date of the call options are set to match the maturity date of the digital option.

- Only at-the-money call options are traded, i.e. the strike K_i is chosen to be the current stock value S_i.

Denoting the portfolio updated at time t_i by $p(t_i)$, the number of call options traded at time t_i by $\#_c(t_i)$ and the number of stocks in the portfolio at time t_i by $\#_S(t_i)$, the updating is performed according to

$$\#_c(t_{i+1}) = \frac{\zeta_d(t_{i+1}, S(t_{i+1}), V(t_{i+1})) - \zeta_{p(t_i)}(t_{i+1}, S(t_{i+1}), V(t_{i+1}))}{\zeta_c(t_{i+1}, S(t_{i+1}), V(t_{i+1}))},$$

$$\begin{aligned}\#_S(t_{i+1}) = \Delta_d(t_{i+1}, S(t_{i+1}), V(t_{i+1})) &- \Delta_{p(t_i)}(t_{i+1}, S(t_{i+1}), V(t_{i+1})) \\ &- \#_c(t_{i+1})\Delta_c(t_{i+1}, S(t_{i+1}), V(t_{i+1})).\end{aligned}$$

3.5 Transaction cost

When trading assets on a real market there always exist a *price spread*, i.e. a difference between the buy and sell price. For stocks this spread is manageable, but when trading options the price spread is considerably large. An appropriate approximation for this spread is the price difference between the option to be traded and an identical option with a BS implied volatility 1% higher, i.e.

$$\Pi_{spread}(c) = \Pi_c(\sigma_{BS} + 0.01) - \Pi_c(\sigma_{BS}).$$

When having derived a digital option Vega-hedge strategy one can now calculate the transaction cost associated with the call option trading.

There are two major factors that affecting the transaction cost:

1. *Vega limits;* The portfolio Vega is updated only if it exceeds this limit.
2. *Trading dates;* Opportunities for trading to attain a Delta- and Vega-neutral portfolio (provided that the Vega limit is exceeded).

One can now incorporate this transaction cost when updating the portfolio as

$$\begin{aligned}\Pi_{p(t_i)} = \Pi_{p(t_{i-1})}e^{r(t_i - t_{i-1})} &+ S(t_{i-1})\Delta_{p(t_{i-1})}(e^{d(t_i - t_{i-1})} - 1) \\ &- S(t_i)\left(\#_S(t_i) - \#_S(t_{i-1})\right) \\ &- \#_c(t_i)\Pi_c(t_i) \\ &- \Pi_{spread}(c) \mid \#_c(t_i) \mid,\end{aligned}$$

where the terms represent the interest capital growth, dividends, stock trading cost, call option trading cost and the Vega-hedging transaction cost, respectively. At

the maturity time T the portfolio value decreases by the digital option payoff and increases by the call option payoffs. By iterating and calculating the discounted mean, the digital option price including the transaction cost is attained. This is done for 10000 iterations and the Vega limit 0.1, similar to the digital option Vega on the signing day, yielding the results:

- Transaction cost$= 13,0\%$, when trading once a day.
- Transaction cost$= 9,0\%$, when trading once a week.

A Additional Plots

A.1 Convergence results

A.1.1 Price

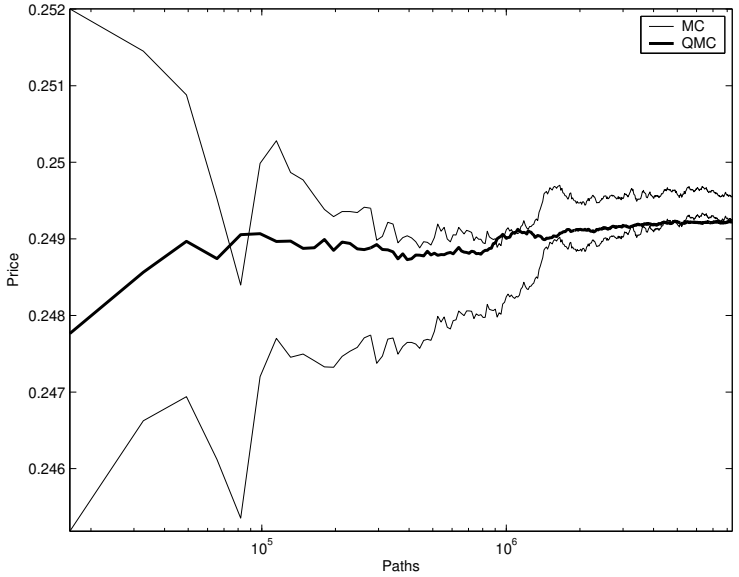


Figure 3: Price of Triple Asian-Digital

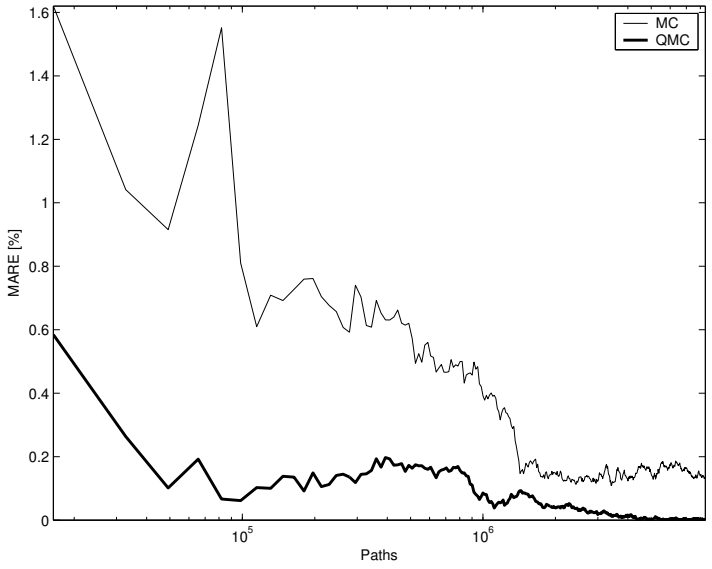


Figure 4: MARE of Triple Asian-Digital price

A.1.2 Delta

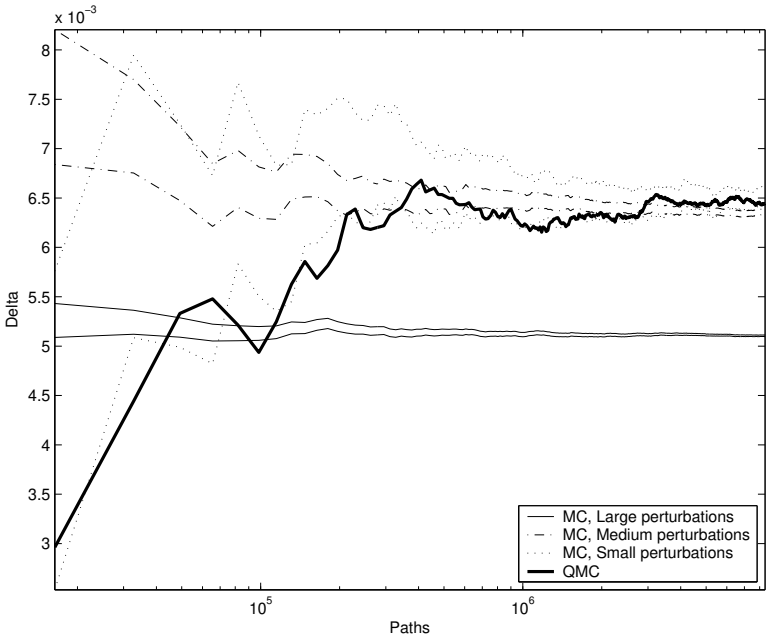


Figure 5: Delta of Triple Asian-Digital

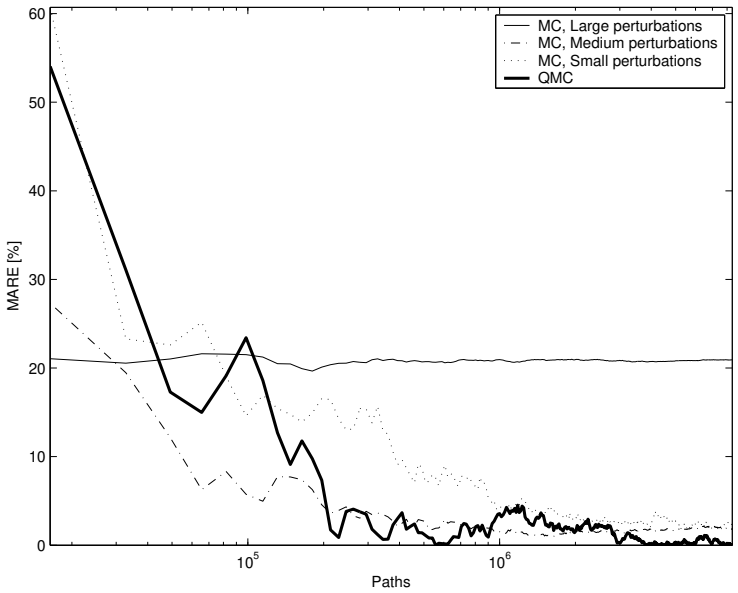


Figure 6: MARE of Triple Asian-Digital Delta

A.1.3 Gamma

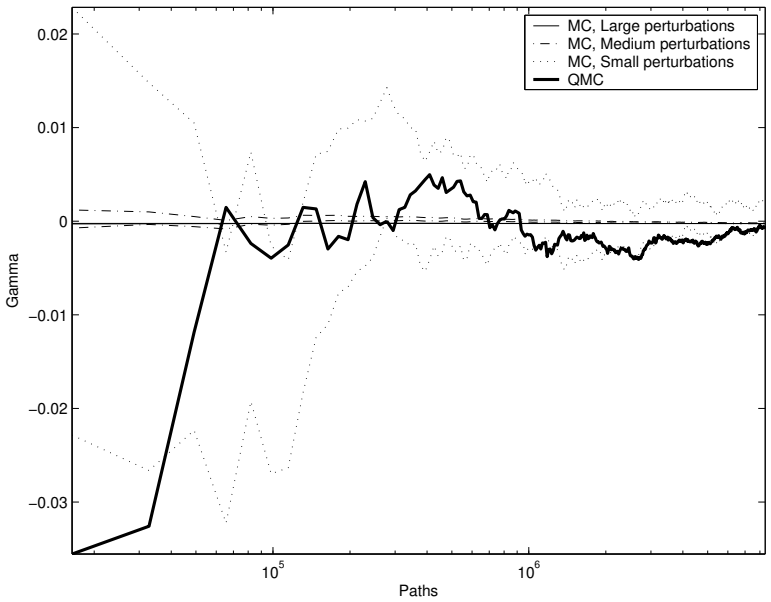


Figure 7: Gamma of Triple Asian-Digital

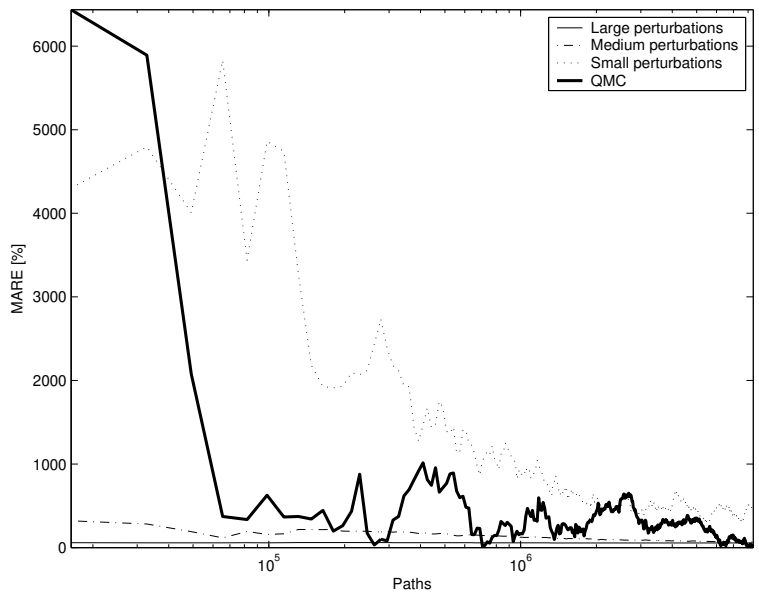


Figure 8: MARE of Triple Asian-Digital Gamma

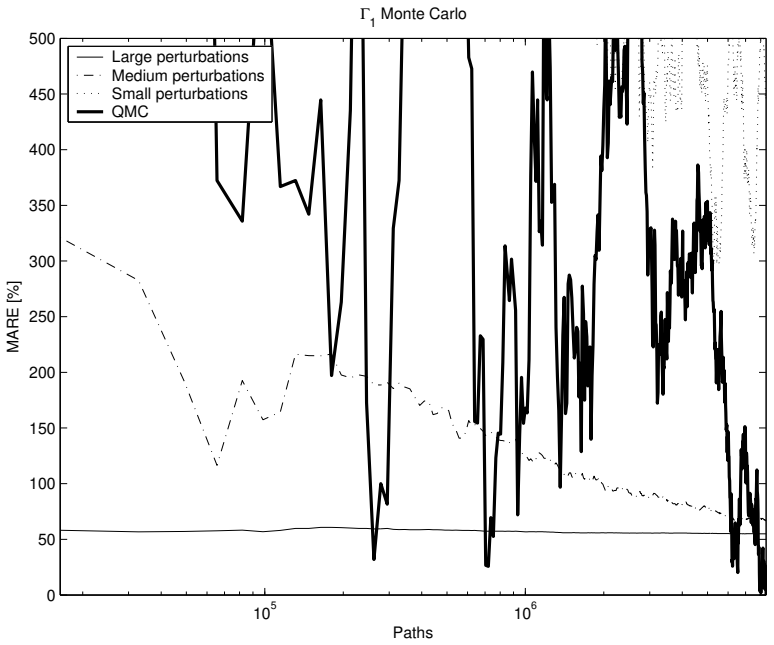


Figure 9: MARE of Triple Asian-Digital Gamma

A.1.4 Cross-Gamma

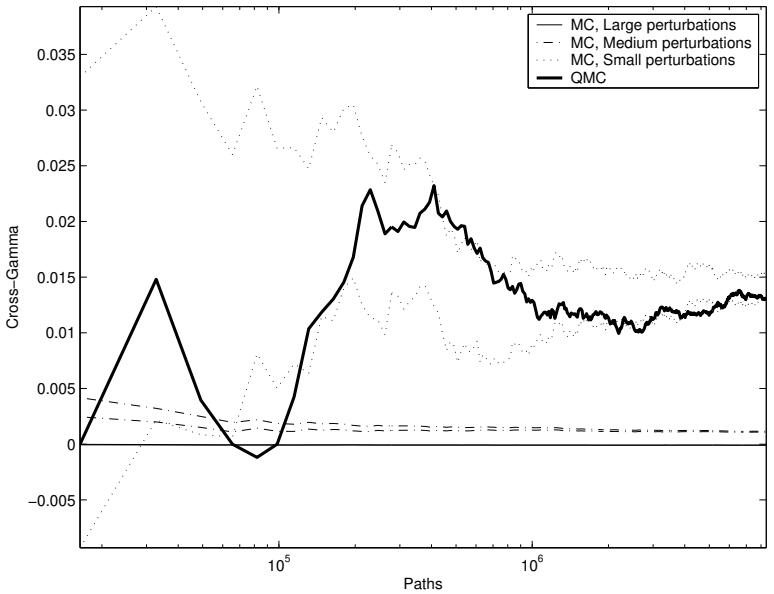


Figure 10: Cross-Gamma of Triple Asian-Digital

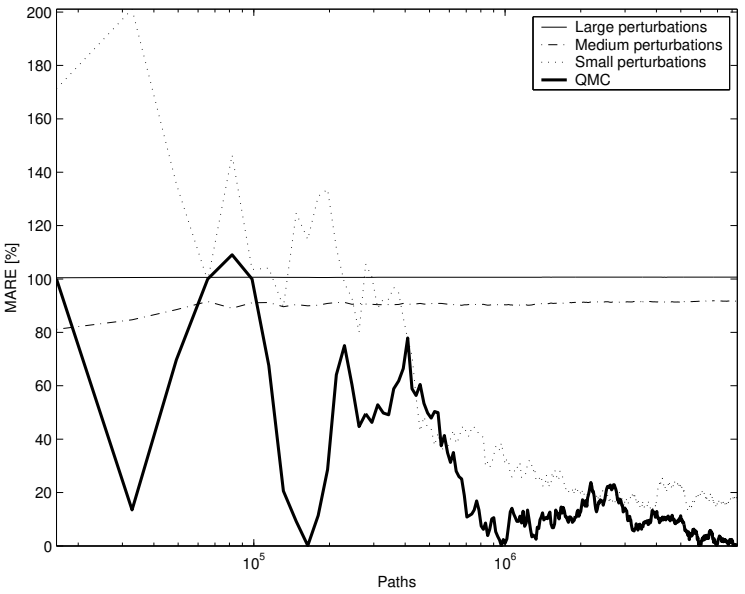


Figure 11: MARE of Triple Asian-Digital Cross-Gamma

A.1.5 Vega

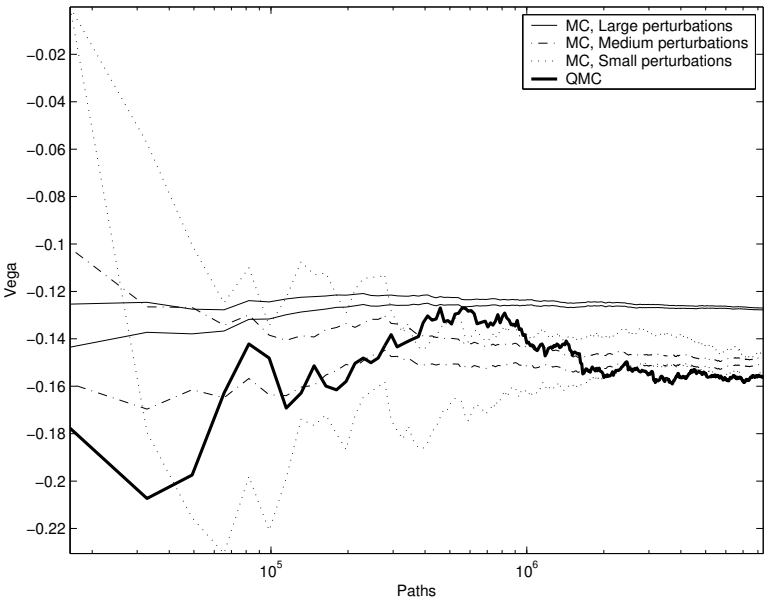


Figure 12: Vega of Triple Asian-Digital

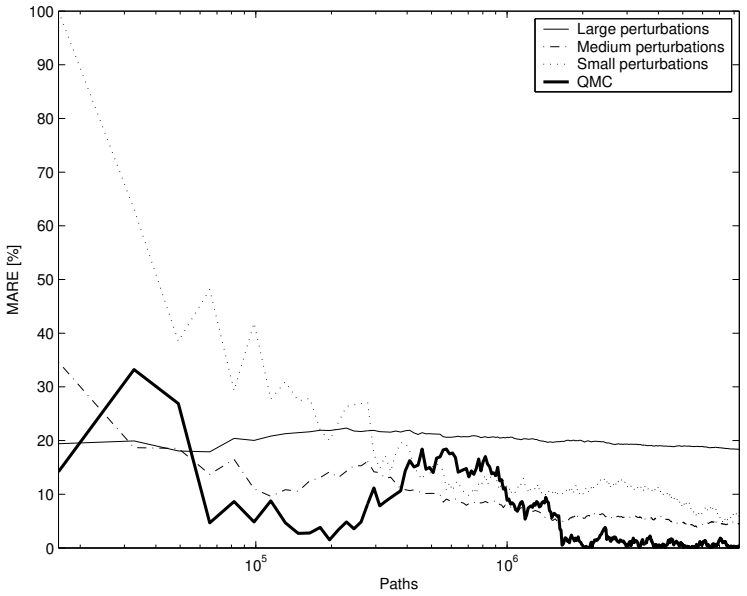


Figure 13: MARE of Triple Asian-Digital Vega

A.1.6 Vomma

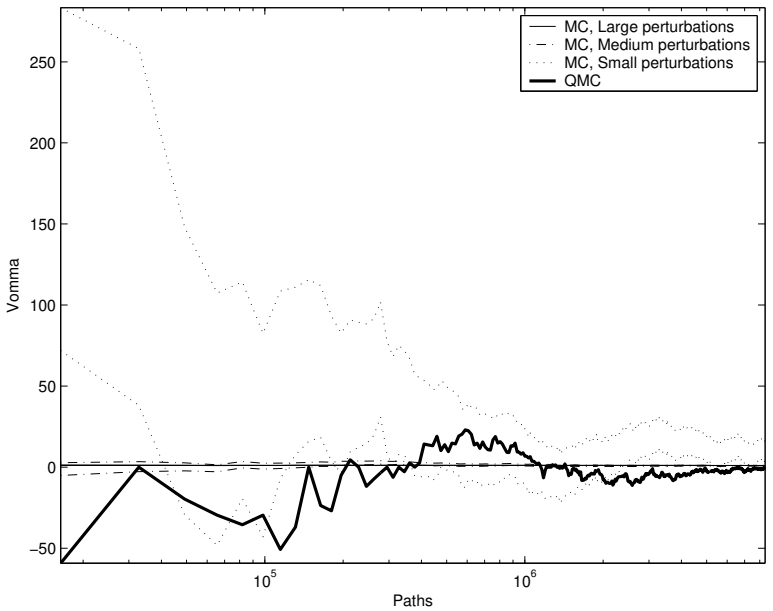


Figure 14: Vomma of Triple Asian-Digital

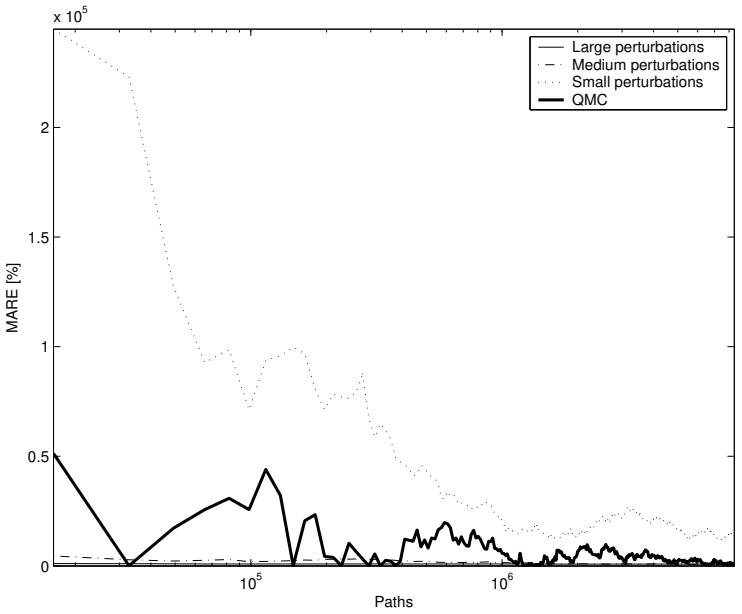


Figure 15: MARE of Triple Asian-Digital Vomma

A.1.7 Theta

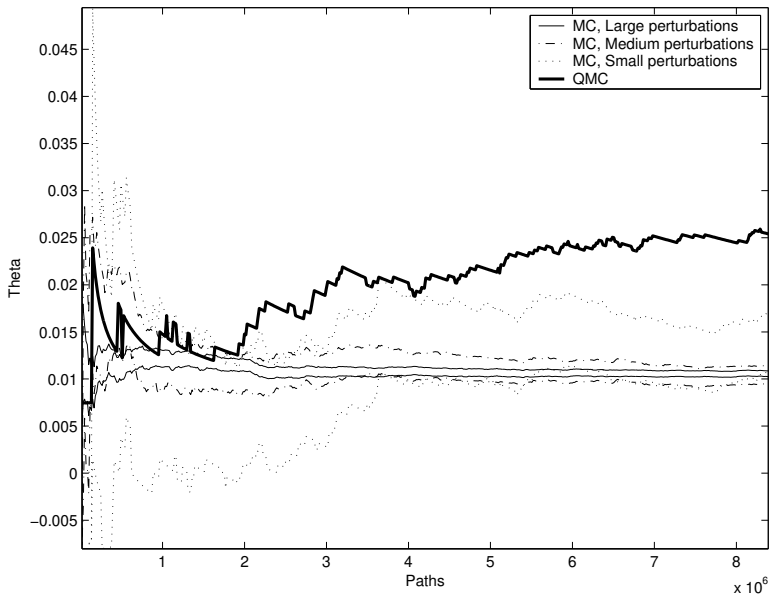


Figure 16: Theta of Triple Asian-Digital

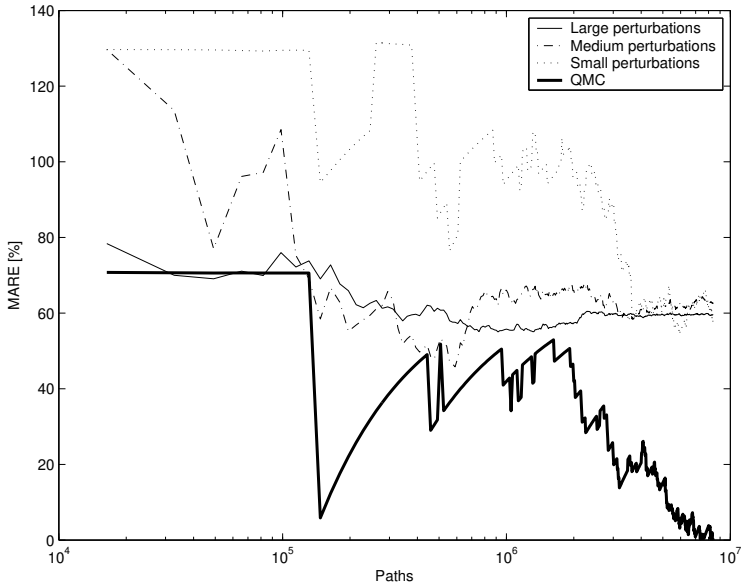


Figure 17: MARE of Triple Asian-Digital Theta

A.1.8 Correlation risk

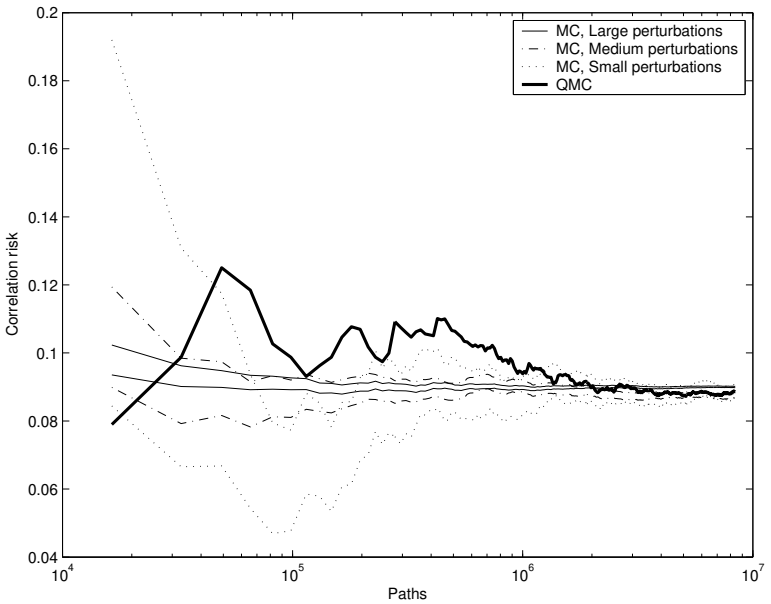


Figure 18: Correlation risk of Triple Asian-Digital

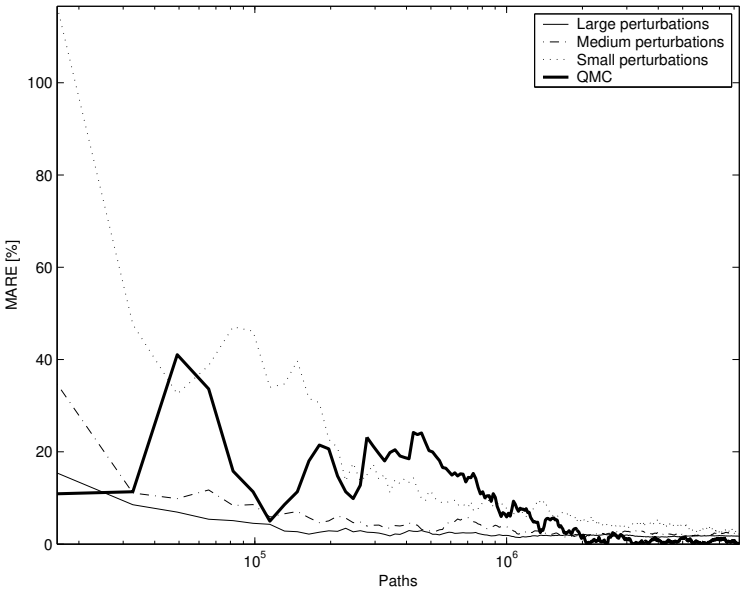


Figure 19: MARE of Triple Asian-Digital Correlation risk

B MATLAB Code

```
function y=MonteCarlo()

Increment=2^14;
Paths=Increment:Increment:2^23;
MCPrice=zeros(length(Paths),19);
MCInterval=zeros(length(Paths),19);
%--------------------"Medium perturbations--------------------------------
dS=1;
ds=0.01;
dT=1;
drho=0.01;
%-------------------------------------------------------------------------
%-----------------------Correlation---------------------------------------
rho12=0.3;
rho13=0.4;
rho23=0.5;
d22=sqrt(1-rho12^2);
d32=(rho23 -rho12*rho13)/d22;
CorrelationCoefficients=...
    [1 0 0; rho12 d22 0; rho13 d32 sqrt(1-rho13^2-d32^2)];
%-------------------------------------------------------------------------
%----------------Perturbations in correlation-----------------------------
rho12=rho12+drho;
rho13=rho13+drho;
rho23=rho23+drho;
d22=sqrt(1-rho12^2);
d32=(rho23 -rho12*rho13)/d22;
CorrelationRiskCoefficients=...
    [1 0 0; rho12 d22 0; rho13 d32 sqrt(1-rho13^2-d32^2)];
%-------------------------------------------------------------------------
%--------------Volatility-------------------------------------------------
Volatility=diag([0.2 0.25 0.3]);
%-------------------------------------------------------------------------
%---------------Stock prices at time t0-----------------------------------
S0=[100,100,100]';
%-------------------------------------------------------------------------
%-------------Interest----------------------------------------------------
r=0.03;
%-------------------------------------------------------------------------
%--------------Timestep---------------------------------------------------
```

```
dt=[231 ones(1,19)]/250;
t=cumsum(dt);
%-------------------------------------------------------------------------
%-------------Perturbation in timestep-----------------------------------
dtTheta=[231-dT ones(1,19)]/250;
tTheta=cumsum(dtTheta);
%-------------------------------------------------------------------------
%-------------Perturbation in timestep-----------------------------------
dtDTheta=[231+dT ones(1,19)]/250;
tDTheta=cumsum(dtDTheta);
%-------------------------------------------------------------------------
%--------------Deterministic part of trajectories-------------------------
ST=S0*ones(1,20).*exp((r*ones(3,1)-diag(Volatility).^2/2)*t);
DELTA=(S0+dS*ones(3,1))*ones(1,20)...
    .*exp((r*ones(3,1)-diag(Volatility).^2/2)*t);
GAMMA=(S0-dS*ones(3,1))*ones(1,20)...
    .*exp((r*ones(3,1)-diag(Volatility).^2/2)*t);
VEGA=S0*ones(1,20).*exp((r*ones(3,1)-diag(Volatility+ds*eye(3)).^2/2)*t);
VOMMA=S0*ones(1,20).*exp((r*ones(3,1)-diag(Volatility-ds*eye(3)).^2/2)*t);
THETA=S0*ones(1,20).*exp((r*ones(3,1)-diag(Volatility).^2/2)*tTheta);
DTHETA=S0*ones(1,20).*exp((r*ones(3,1)-diag(Volatility).^2/2)*tDTheta);
%-------------------------------------------------------------------------
%--------------Iteration--------------------------------------------------
Price=zeros(1,19);
PriceSquared=zeros(1,19);
CrossPrice=zeros(1,19);
CrossCrossPrice=zeros(1,19);
FirstOrderStd=zeros(length(Paths),19);
SecondOrderStd=zeros(length(Paths),19);

for j=1:length(Paths)
    for i=1:Increment
        %-------------Correlated Brownian motion-------------------------
        N=randn(3,20);
        Z=...%Stochastic part of trajectories
            CorrelationCoefficients*cumsum(ones(3,1)*sqrt(dt).*N,2);
        ZTheta=...%Perturbated in time
            CorrelationCoefficients*cumsum(ones(3,1)*sqrt(dtTheta).*N,2);
        ZDTheta=...%Perturbated in time
            CorrelationCoefficients*cumsum(ones(3,1)*sqrt(dtDTheta).*N,2);
        ZCorrelationRisk=...%Perturbated in correlation
            CorrelationRiskCoefficients*cumsum(ones(3,1)*sqrt(dt).*N,2);
```

```
    %-----------------------------------------------------------------
    %----------------Trajectories-------------------------------------
    St=ST.*exp(Volatility*Z);
    Delta=DELTA.*exp(Volatility*Z);
    Gamma=GAMMA.*exp(Volatility*Z);
    Vega=VEGA.*exp((Volatility+ds*eye(3))*Z);
    Vomma=VOMMA.*exp((Volatility-ds*eye(3))*Z);
    Theta1=THETA.*exp(Volatility*ZTheta);
    DTheta1=DTHETA.*exp(Volatility*ZDTheta);
    CorrelationRisk1=ST.*exp(Volatility*ZCorrelationRisk);
    %-----------------------------------------------------------------
    %----------Payoff-------------------------------------------------
    Trash=[Evaluate(St,S0)...
            Evaluate([Delta(1,:);St(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);Delta(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);St(2,:);Delta(3,:)],S0)...
            Evaluate([Gamma(1,:);St(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);Gamma(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);St(2,:);Gamma(3,:)],S0)...
            Evaluate([Vega(1,:);St(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);Vega(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);St(2,:);Vega(3,:)],S0)...
            Evaluate([Vomma(1,:);St(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);Vomma(2,:);St(3,:)],S0)...
            Evaluate([St(1,:);St(2,:);Vomma(3,:)],S0)...
            exp(-r*(dT/250))*Evaluate(Theta1,S0)...
            exp(r*(dT/250))*Evaluate(DTheta1,S0)...
            Evaluate(CorrelationRisk1,S0)...
            0 0 0];
    %-----------------------------------------------------------------
    %------Calculations for sample standard deviation-----------------
    Price=Price+Trash;
    PriceSquared=PriceSquared +Trash.^2;
    CrossPrice=CrossPrice+Evaluate(St,S0).*Trash;
    CrossCrossPrice=CrossCrossPrice...
        +[0 0 0 0 Trash(2:4).*Trash(5:7) 0 0 0 ...
            Trash(8:10).*Trash(11:13) 0 Trash(14)*Trash(15)...
            0 Trash([2,2,3]).*Trash([6,7,7])];
    %-----------------------------------------------------------------
end
%--------Sample standard deviation--------------------------------------
FirstOrderStd(j,:)=((PriceSquared+PriceSquared(1)-2*CrossPrice)...
```

```
    /Paths(j)-2*(Price.^2+Price(1).^2-2*Price.*Price(1))/Paths(j).^2);

SecondOrderStd(j,5:7)=((PriceSquared(2:4)+4*PriceSquared(1)...
    +PriceSquared(5:7)-4*CrossPrice(2:4)-4*CrossPrice(5:7)...
    +2*CrossCrossPrice(5:7))/Paths(j)...
    -(Price(2:4).^2+4*Price(1).^2+Price(5:7).^2 ...
    -4*Price(1)*Price(2:4)-4*Price(1)*Price(5:7)...
    +2*Price(2:4).*Price(5:7))/Paths(j).^2);

SecondOrderStd(j,11:13)=((PriceSquared(8:10)+4*PriceSquared(1)...
    +PriceSquared(11:13)-4*CrossPrice(8:10)-4*CrossPrice(11:13)...
    +2*CrossCrossPrice(11:13))/Paths(j)...
    -(Price(8:10).^2+4*Price(1).^2+Price(11:13).^2 ...
    -4*Price(1)*Price(8:10)-4*Price(1)*Price(11:13)...
    +2*Price(8:10).*Price(11:13))/Paths(j).^2);

SecondOrderStd(j,15)=((PriceSquared(14)+4*PriceSquared(1)...
    +PriceSquared(15)-4*CrossPrice(14)-4*CrossPrice(15)...
    +2*CrossCrossPrice(15))/Paths(j)...
    -(Price(14).^2+4*Price(1).^2+Price(15).^2 ...
    -4*Price(1)*Price(14)-4*Price(1)*Price(15)...
    +2*Price(14).*Price(15))/Paths(j).^2);

SecondOrderStd(j,17:19)=((PriceSquared([6,7,7])+4*PriceSquared(1)...
    +PriceSquared([2,2,3])-4*CrossPrice([6,7,7])...
    -4*CrossPrice([2,2,3])+2*CrossCrossPrice(17:19))/Paths(j)...
    -(Price([6,7,7]).^2+4*Price(1).^2+Price([2,2,3]).^2 ...
    -4*Price(1)*Price([6,7,7])-4*Price(1)*Price([2,2,3])...
    +2*Price([6,7,7]).*Price([2,2,3]))/Paths(j).^2);

MCInterval(j,:)=1/sqrt(Paths(j))*sqrt(FirstOrderStd(j,:))...
    ./[1 dS*ones(1,6) ds*ones(1,6) dT/250 dT/250 3*drho dS dS dS];
MCInterval(j,1)=1/sqrt(Paths(j))...
    *sqrt(PriceSquared(1)/Paths(j)-Price(1).^2/Paths(j).^2);
MCInterval(j,5:7)=1/sqrt(Paths(j))...
    *sqrt(SecondOrderStd(j,5:7))/dS^2;
MCInterval(j,11:13)=1/sqrt(Paths(j))...
    *sqrt(SecondOrderStd(j,11:13))/ds^2;
MCInterval(j,15)=1/sqrt(Paths(j))...
    *sqrt(SecondOrderStd(j,15))/(dT/250)^2;
MCInterval(j,17:19)=1/sqrt(Paths(j))...
    *sqrt(SecondOrderStd(j,17:19))/dS^2;
```

```
    %-------------------------------------------------------------------
    %-----------------Price and Greeks----------------------------------
    MCPrice(j,1)=exp(-r)*Price(1)/Paths(j);
    MCPrice(j,2:4)=exp(-r)*(Price(2:4)-Price(1))/Paths(j)/dS;
    MCPrice(j,5:7)=exp(-r)*(Price(5:7)-2*Price(1)+Price(2:4))...
        /Paths(j)/dS^2;
    MCPrice(j,8:10)=exp(-r)*(Price(8:10)-Price(1))/Paths(j)/ds;
    MCPrice(j,11:13)=exp(-r)*(Price(11:13)-2*Price(1)+Price(8:10))...
        /Paths(j)/ds^2;
    MCPrice(j,14)=exp(-r)*(Price(14)-Price(1))/Paths(j)/(dT/250);
    MCPrice(j,15)=exp(-r)*(Price(15)-2*Price(1)+Price(14))/Paths(j)...
        /(dT/250)^2;
    MCPrice(j,16)=exp(-r)*(Price(16)-Price(1))/Paths(j)/(3*drho);
    MCPrice(j,17:19)=exp(-r)*(Price([2,2,3])-2*Price(1)+Price([6,7,7]))...
        /Paths(j)/dS^2;
    %-------------------------------------------------------------------
end
%--------------Plots-----------------------------------------------------
titlar=['\Pi Monte Carlo                  ';...
        '\Delta_1 Monte Carlo             ';...
        '\Delta_2 Monte Carlo             ';...
        '\Delta_3 Monte Carlo             ';...
        '\Gamma_1 Monte Carlo             ';...
        '\Gamma_2 Monte Carlo             ';...
        '\Gamma_3 Monte Carlo             ';...
        '\Pi_{\sigma_1} Monte Carlo       ';...
        '\Pi_{\sigma_2} Monte Carlo       ';...
        '\Pi_{\sigma_3} Monte Carlo       ';...
        '\Pi_{\sigma_1\sigma_1} Monte Carlo';...
        '\Pi_{\sigma_2\sigma_2} Monte Carlo';...
        '\Pi_{\sigma_3\sigma_3} Monte Carlo';...
        '\Theta Monte Carlo               ';...
        '\Pi_{tt} Monte Carlo             ';...
        '\Pi_{\rho} Monte Carlo           ';...
        '\Gamma_{12} Monte Carlo          ';...
        '\Gamma_{13} Monte Carlo          ';...
        '\Gamma_{23} Monte Carlo          ';...
];
titlar=cellstr(titlar);
for j=1:19
    figure(j)
    hold on
```

```
    p1=plot(Paths,[MCPrice(:,j)-MCInterval(:,j)],':k')
    hold on
    p2=plot(Paths,[MCPrice(:,j)+MCInterval(:,j)],':k')
    hold on
    p3=plot(Paths,MCPrice(:,j),'k')
    legend([p1 p3],'MC sample standard deviation interval','MC Sample')
    title(titlar(j))
    xlabel('Paths')
    ylabel(titlar(j))
end
%-------------------------------------------------------------------------
y=0;


function y=Evaluate(S,S0) %Payoff function for Triple Digital-Asian
y1=0;
R1=sort(S,2);
if min(sign(mean(R1(:,11:20)')'-S0))>=0
    y1=1;
end
y=y1;


function y=ApproximationByTripleDigital()
%------Initial guess-----------------------------------
K=1;
t=1;
%-----------------------------------------------------------
x0=[t,K];
lb=[0.5 0.5];
ub=[1.5 1.5];
[x,resnorm,residual,exitflag,output,lambda,jacobian]=...
    lsqnonlin(@PenaltyFunction,x0,lb,ub)
y=x


function y=TrippeldigitalPV(r,t,y12,y13,y23,S0,v,K)
%--------------Parameters-------------------------------------------------
s1=S0(1);
s2=S0(2);
s3=S0(3);
v1=v(1);
```

```
v2=v(2);
v3=v(3);
x1=K(1);
x2=K(2);
x3=K(3);
q1=0;
q2=0;
q3=0;
h1=(r-q1-v1^2/2)*t;
g1=v1*sqrt(t);
x21=y12;
x22=sqrt(1-x21^2);
x31=y13;
x32=(y23-y12*y13)/x22;
x33=sqrt(1-x31^2-x32^2);
e=0.00000001;
p10=-(log(s1/x1)+h1)/g1;
p11=sqrt(-2*log(e*sqrt(2*pi)*g1));
df=exp(-r*t);
%-------------------------------------------------------------------------
%--------------Integral---------------------------------------------------
n=24;
d1=(p11-p10)/n;
sum1=0;
for i1=0:n
    p1=p10+i1*d1;
    h2=(r-q2-v2^2/2)*t+v2*sqrt(t)*x21*p1;
    g2=v2*sqrt(t)*x22;
    p20=-(log(s2/x2)+h2)/g2;
    p21=sqrt(-2*log(e*sqrt(2*pi)*g2));
    d2=(p21-p20)/n;
    sum2=0;
    for i2=0:n
        p2=p20+i2*d2;
        h3=(r-q3-v3^2/2)*t+v3*sqrt(t)*(x31*p1+x32*p2);
        g3=v3*sqrt(t)*x33;
        p30=-(log(s3/x3)+h3)/g3;
        p31=sqrt(-2*log(e*sqrt(2*pi)*g3));
        k2=exp(-p2^2/2)*normcdf(-p30,0,1);
        if i2==0 | i2==n
            sum2=sum2+k2;
        elseif mod(i2,2)==0
```

```
            sum2=sum2+2*k2;
        else
            sum2=sum2+4*k2;
        end
    end
    k1=exp(-p1^2/2)/sqrt(2*pi)*d2*sum2/3;
    if i1==0 | i1==n
        sum1=sum1+k1;
    elseif mod(i1,2)==0
        sum1=sum1+2*k1;
    else
        sum1=sum1+4*k1;
    end
end
y=df/sqrt(2*pi)*d1*sum1/3;
%-------------------------------------------------------------------------


function y=PenaltyFunction(x)
t=x(1);
K=x(2);
%---------------Calibration targets-----------------------------------------
V=0.24872608908338;
Delta1=0.00651544242495;
Delta2=0.00534007141931;
Delta3=0.00425724931176;
Vega1=-0.14992921489225;
Vega2=-0.11846258954450;
Vega3=-0.09810183196654;
CorrelationRisk=0.08668746786980;
%-------------------------------------------------------------------------
%----------Std for Calibration targets-----------------------------------
c(1)=0.00042635734731;
c(2)=0.00025286844669;
c(3)=0.00022895434112;
c(4)=0.00020445060414;
c(5)=0.01213641691787;
c(6)=0.01078827604937;
c(7)=0.00981768542147;
c(8)=0.00536520699098;
%-------------------------------------------------------------------------
%-------------- Correlation coefficients-----------------------------------
```

```
rho12=0.3;
rho13=0.4;
rho23=0.5;
%-------------------------------------------------------------------------
%--------------Volatility-------------------------------------------------
v(1)=0.25;
v(2)=0.3;
v(3)=0.35;
%-------------------------------------------------------------------------
%---------------Initial stock prices--------------------------------------
S0(1)=100;
S0(2)=100;
S0(3)=100;
%-------------------------------------------------------------------------
%--------------interest---------------------------------------------------
r=0.03;
%-------------------------------------------------------------------------
%---------------------Perturbations---------------------------------------
dS=1;
dv=0.01;
dt=1/250;
drho=0.01;
%-------------------------------------------------------------------------
%-------------------Perturbations-----------------------------------------
b(1)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0,v,K*S0);
b(2)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0+[dS 0 0],v,K*S0);
b(3)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0+[0 dS 0],v,K*S0);
b(4)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0+[0 0 dS],v,K*S0);
b(5)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0,v+[dv 0 0],K*S0);
b(6)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0,v+[0 dv 0],K*S0);
b(7)=TrippeldigitalPV(r,t,rho12,rho13,rho23,S0,v+[0 0 dv],K*S0);
b(8)=TrippeldigitalPV(r,t,rho12+drho,rho13+drho,rho23+drho,S0,v,K*S0);
%-------------------------------------------------------------------------
%------------------Price and Greeks---------------------------------------
b(1)=b(1);
b(2)=(b(2)-b(1))/dS;
b(3)=(b(3)-b(1))/dS;
b(4)=(b(4)-b(1))/dS;
b(5)=(b(5)-b(1))/dv;
b(6)=(b(6)-b(1))/dv;
b(7)=(b(7)-b(1))/dv;
b(8)=(b(8)-b(1))/(3*drho);
```

```
%-------------------------------------------------------------------------
y(1)=(b(1)-V)/c(1);
y(2)=(b(2)-Delta1)/c(2);
y(3)=(b(3)-Delta2)/c(3);
y(4)=(b(4)-Delta3)/c(4);
y(5)=(b(5)-Vega1)/c(5);
y(6)=(b(6)-Vega2)/c(6);
y(7)=(b(7)-Vega3)/c(7);
y(8)=(b(8)-CorrelationRisk)/c(8);
y=y.^2;

function y=BatesCalibration()
%----------Parameters-------------------------------------------------------
tau=1;
r=0.0224;
d=0.0295;
S=100;
%-------------------------------------------------------------------------
%--------Initial guess----------------------------------------------------
sigma=0.1325;
V=sigma^2;
kappa=10;
theta=0.1325^2;
epsilon=0.5;
rho=-0.5;
lambda=0.3635;
ny=-0.2459;
delta=0.2547;
x0=[V,kappa,theta,epsilon,rho,lambda,ny,delta];
%-------------------------------------------------------------------------
%---------Optimization----------------------------------------------------
[x,fval,exitflag,output]=fminsearch(@PenaltyFunction,x0,...
   optimset('Display','notify'))
%-------------------------------------------------------------------------
y=0;


function y=PenaltyFunction(x)
%----------parameters-------------------------------------------------------
tau=1;
r=0.0224;
d=0.0295;
```

```
S=100;
%-------------------------------------------------------------------------
%-----------Target values-------------------------------------------------
percent=[90:0.2:110];
K=percent/100*S;
sigma=0.1325; %at-the-money
sigma=sigma+0.002*(100-percent); %skew
weights=abs((100-percent).^2-100);
%-------------------------------------------------------------------------
%-----Least-square penalty function-----------------------------------------
sum=0;
for i=1:length(percent)
    sum=sum+weights(i)...
        .*(BatesImpliedVolatility(x,tau,S,K(i),r,d)-sigma(i)).^2;
end
%-------------------------------------------------------------------------
y=sum;


function y=BatesImpliedVolatility(x,tau,S,K,r,d)
j=1;
Integral=0;
inc=quadl(@Integrand,0,1,[],[],x,tau,S,K,r,d);
while abs(inc)>1e-8
    Integral=Integral+inc;
    inc=quadl(@Integrand,j,j+1,[],[],x,tau,S,K,r,d);
    j=j+1;
end
Integral=Integral+inc;
CallPrice=S*exp(-tau.*d)-K/pi*Integral;
y=ImpliedVolatilityBisections(S,K,r,d,tau,CallPrice);


function y=Integrand(k,x,tau,S,K,r,d)
V=x(1);
kappa=x(2);
theta=x(3);
epsilon=x(4);
rho=x(5);
lambda=x(6);
ny=x(7);
delta=x(8);
```

```
X=log(S./K)+(r-d).*tau;
u=kappa-rho.*epsilon./2;
xi=sqrt(k.^2.*epsilon.^2.*(1-rho.^2)...
    +2.*i.*k.*rho.*epsilon.*u +u.^2+epsilon.^2./4);
PsiPlus=-(u+i.*k.*rho.*epsilon)+xi;
PsiMinus=(u+i.*k.*rho.*epsilon)+xi;
A=-(kappa.*theta./(epsilon.^2))...
    .*(PsiPlus.*tau+2.*log((PsiMinus+PsiPlus.*exp(-xi.*tau))./(2.*xi)));
B=-(k.^2+1/4).*(1-exp(-xi.*tau))./(PsiMinus+PsiPlus.*exp(-xi.*tau));
LAMBDA=exp(-i.*k.*(ny+delta.^2./2)-(k.^2-1./4).*delta.^2./2+ny./2)...
    -1-(-i.*k+1./2).*(exp(ny+delta.^2./2)-1);
D=tau.*LAMBDA;
Q=exp((-i.*k+1./2).*X+A+B.*V...
    +D.*lambda);
y=real(Q./(k.^2+1/4));

function y=ImpliedVolatilityBisections(S,K,r,d,tau,option_price)
Y1=0;
Y2=0;
Y3=0;
Y4=0;
ACCURACY=1e-6;
MAX_ITERATIONS=200;
HIGH_VALUE=1e10;
ERR=-1e40;
%------arbitrage test-----------------------------------------------------
sigma_low=0.0001;
price=BlackScholesEuro(S,K,r,d,tau,sigma_low);
if(price>option_price)
    Y1=ERR;
end
%-------------------------------------------------------------------------
sigma_high=0.3;
price=BlackScholesEuro(S,K,r,d,tau,sigma_high);
while(price<option_price)
    sigma_high=2*sigma_high;
    price=BlackScholesEuro(S,K,r,d,tau,sigma_high);
    if(sigma_high>HIGH_VALUE)
        Y2=ERR;
    end
end
for i=0:MAX_ITERATIONS
```

```
    sigma=(sigma_low+sigma_high)*0.5;
    price=BlackScholesEuro(S,K,r,d,tau,sigma);
    test=(price-option_price);
    if(abs(test)<ACCURACY)
        Y3=sigma;
        break
    end
    if(test<0)
        sigma_low=sigma;
    else
        sigma_high=sigma;
    end
    if(i==MAX_ITERATIONS)
        Y4=ERR;
    end
end
if (Y1~=0)
    y=Y1;
elseif (Y2~=0)
    y=Y2;
elseif(Y4~=0)
    y=Y4;
else
    y=Y3;
end

function y=BlackScholesEuro(S,K,r,d,tau,sigma)
dplus=1/sigma/sqrt(tau)*(log(S/K)+(r-d+sigma^2/2)*tau);
dminus=1/sigma/sqrt(tau)*(log(S/K)+(r-d-sigma^2/2)*tau);
y=S*exp(-d*tau)*normcdf(dplus)-K*exp(-r*tau)*normcdf(dminus);


function y=BatesVegaHedge()
VegaLimit=input('Vega limit= ');
Iterations=input('Iterations= ');
SpreadVolatility=input('Spread volatility= ');
TradingDayInterval=input('Trading day interval= ');
%-------------Results from Calibration-------------------------------------
V0=0.0182;
kappa=10.0077;
theta=0.0149;
epsilon=0.4731;
```

```
rho=-0.4921;
lambda=0.3398;
ny=-0.2090;
delta=0.2553;
%-------------------------------------------------------------------------
%------Parameters-------------------------------------------------------
x=[kappa,theta,epsilon,rho,lambda,ny,delta];
BSVolatility=sqrt(V0);
r=0.0224;
d=0.0295;
S0=100;
K=100;
timesteps=250;
dt=1/timesteps;
tau=1:-dt:0;
FinalTradingDay=245; %poor FFT-results close to maturity
%-------------------------------------------------------------------------
%---------------Trajectories----------------------------------------------
S=zeros(Iterations,timesteps+1);
V=zeros(Iterations,timesteps+1);
S(:,1)=S0*ones(Iterations,1);
V(:,1)=V0*ones(Iterations,1);
m=exp(ny+1/2*delta^2)-1;
for p=1:timesteps
    df=4*kappa*theta/epsilon^2;
    c1=epsilon^2*(1-exp(-kappa*dt))/(4*kappa);
    c2=V(:,p)*exp(-kappa*dt)/c1;
    P=poissrnd(c2/2);
    V(:,p+1)=c1*chi2rnd(df+2*P);
    dW=(V(:,p+1)-V(:,p)-kappa*(theta-V(:,p))*dt)...
        ./(epsilon*sqrt(V(:,p)));
    dX=...%Euler increment for dV.
        (rho*dW+sqrt(1-rho^2)*sqrt(dt)*randn(Iterations,1));
    S(:,p+1)=S(:,p)+(r-d-lambda*m)*S(:,p)*dt...
        +sqrt(V(:,p)).*S(:,p).*dX...
        +(exp(ny+delta*randn(Iterations,1))-1)...
        .*S(:,p).*poissrnd(dt*lambda,Iterations,1);
end
%-----------FFT parameters-------------------------------------------------
N=4096;
FFTeta=0.25;
FFTlambda=2*pi/N/FFTeta;
```

```
b=N*FFTlambda/2;
k=FFTeta*[0:N-1];
Stock=exp(-b+FFTlambda.*([0:N-1]));
%-------------------------------------------------------------------------
%------Interpolation prerequisites---------------------------------------
MaxVolatility=sqrt(max(max(V)));
Vspread=[0:0.02:MaxVolatility+0.02].^2;
[X Y]=ndgrid(Stock,Vspread);  %For interpolation
SimpsonVector=(2*mod([0:N-1],2)+2)/3; %For Simpson integral approx.
SimpsonVector(1)=1/3;
%-------------------------------------------------------------------------
%---------------FFT pricing-----------------------------------------------
for p=[1:(length(tau)-1)]
    [p]
    FFTCallPrice=exp(-i.*k'*ones(1,length(Vspread))...
        .*((r-d)*tau(p)-log(K))).*QStar(k'*ones(1,length(Vspread))...
        ,ones(length(k),1)*Vspread,x,tau(p),r,d)...
        ./(k'.^2*ones(1,length(Vspread))+1/4)...
        .*(SimpsonVector'*ones(1,length(Vspread)));
    FFTCallDelta=exp(-i.*k'*ones(1,length(Vspread))...
        .*((r-d)*tau(p)-log(K))).*QStar(k'*ones(1,length(Vspread))...
        ,ones(length(k),1)*Vspread,x,tau(p),r,d)...
        ./(i.*k'*ones(1,length(Vspread))+1/2)...
        .*(SimpsonVector'*ones(1,length(Vspread)));
    FFTCallVega=exp(-i.*k'*ones(1,length(Vspread))...
        .*((r-d)*tau(p)-log(K))).*B(k'*ones(1,length(Vspread))...
        ,x,tau(p),r,d)...
        .*QStar(k'*ones(1,length(Vspread))...
        ,ones(length(k),1)*Vspread,x,tau(p),r,d)...
        ./(k'.^2*ones(1,length(Vspread))+1/4)...
       *(SimpsonVector'*ones(1,length(Vspread)));
%     FFTDigitalPrice=exp(-i.*k'*ones(1,length(Vspread))...
%         .*((r-d)*tau(p)-log(K))).*QStar(k'*ones(1,length(Vspread))...
%         ,ones(length(k),1)*Vspread,x,tau(p),r,d)...
%         ./(-i.*k'*ones(1,length(Vspread))+1/2)...
%         .*(SimpsonVector'*ones(1,length(Vspread)));
    FFTDigitalDelta=exp(-i.*k'*ones(1,length(Vspread))...
        .*((r-d)*tau(p)-log(K))).*QStar(k'*ones(1,length(Vspread))...
        ,ones(length(k),1)*Vspread,x,tau(p),r,d)...
        .*(SimpsonVector'*ones(1,length(Vspread)));
    FFTDigitalVega=exp(-i.*k'*ones(1,length(Vspread))...
        .*((r-d)*tau(p)-log(K)))...
```

```
        .*B(k'*ones(1,length(Vspread)),x,tau(p),r,d)...
        .*QStar(k'*ones(1,length(Vspread))...
        ,ones(length(k),1)*Vspread,x,tau(p),r,d)...
        ./(-i.*k'*ones(1,length(Vspread))+1/2)...
        .*(SimpsonVector'*ones(1,length(Vspread)));

    FFTCallPrice=fft(exp(i*b*k'*ones(1,length(Vspread)))...
        .*FFTCallPrice*FFTeta);
    FFTCallDelta=fft(exp(i*b*k'*ones(1,length(Vspread)))...
        .*FFTCallDelta*FFTeta);
    FFTCallVega=fft(exp(i*b*k'*ones(1,length(Vspread)))...
        .*FFTCallVega*FFTeta);
%     FFTDigitalPrice=fft(exp(i*b*k'*ones(1,length(Vspread)))...
%         .*FFTDigitalPrice*FFTeta);
    FFTDigitalDelta=fft(exp(i*b*k'*ones(1,length(Vspread)))...
        .*FFTDigitalDelta*FFTeta);
    FFTDigitalVega=fft(exp(i*b*k'*ones(1,length(Vspread)))...
        .*FFTDigitalVega*FFTeta);

    FFTCallPrice=Stock'*ones(1,length(Vspread))*exp(-tau(p).*d)...
        -FFTCallPrice*K/pi...
        .*exp(1/2*(log(Stock'*ones(1,length(Vspread)))...
        -log(K)+(r-d)*tau(p)));
    FFTCallDelta=exp(-tau(p)*d)-FFTCallDelta*K...
        ./(Stock'*ones(1,length(Vspread)))/pi.*...
        exp(1/2*(log(Stock'*ones(1,length(Vspread)))...
        -log(K)+(r-d)*tau(p)));
    FFTCallVega=-2*ones(length(Stock),1)*sqrt(Vspread)...
        .*FFTCallVega*K/pi.*exp(1/2*(log(Stock'*ones(1,length(Vspread)))...
        -log(K)+(r-d)*tau(p)));
%     FFTDigitalPrice=FFTDigitalPrice/pi...
%         .*exp(1/2*(log(Stock'*ones(1,length(Vspread)))...
%         -log(K)+(r-d)*tau(p)));
    FFTDigitalDelta=FFTDigitalDelta./(Stock'*ones(1,length(Vspread)))...
        /pi.*exp(1/2*(log(Stock'*ones(1,length(Vspread)))...
        -log(K)+(r-d)*tau(p)));
    FFTDigitalVega=2*ones(length(Stock),1)*sqrt(Vspread).*FFTDigitalVega...
        /pi.*exp(1/2*(log(Stock'*ones(1,length(Vspread)))...
        -log(K)+(r-d)*tau(p)));
    %-------------------------------------------------------------------
    %-----------At-the-money (ATM)-------------------------------------
    ATMCallPrice(:,p)=S(:,p)./K.*interpn(X,Y,real(FFTCallPrice)...
```

```
    ,K*ones(Iterations,1),V(:,p),'linear');
ATMCallDelta(:,p)=interpn(X,Y,real(FFTCallDelta)...
    ,K*ones(Iterations,1),V(:,p),'linear');
ATMCallVega(:,p)=S(:,p)./K.*interpn(X,Y,real(FFTCallVega)...
    ,K*ones(Iterations,1),V(:,p),'linear');
%-------------------------------------------------------------------
%---------------Call option spread----------------------------------
    for h=1:Iterations
        PriceSpread(h,p)=ImpliedVolatilityBisections(...
            100,100,r,d,tau(p),ATMCallPrice(h,p));
        PriceSpread(h,p)=BlackScholesEuro(100,100,r,d,tau(p)...
            ,PriceSpread(h,p)+SpreadVolatility)-ATMCallPrice(h,p);
    end
%-------------------------------------------------------------------
%--------------Vega hedge-------------------------------------------
if p==1
    CallOptions(:,p)=interpn(X,Y,real(FFTDigitalVega)...
        ,S(:,p),V(:,p),'linear')./ATMCallVega(:,p);
elseif p<=FinalTradingDay
    PortfolioVega(:,p)=sum(CallOptions(:,[1:p-1]).*S(:,[1:p-1])./K.*...
        interpn(X,Y,real(FFTCallVega),S(:,p)*ones(1,p-1).*...
        K./S(:,[1:p-1]),V(:,p)*ones(1,p-1),'linear'),2);

    CallOptions(:,p)=(interpn(X,Y,real(FFTDigitalVega)...
        ,S(:,p),V(:,p),'linear')-PortfolioVega(:,p))./ATMCallVega(:,p);
else
    CallOptions(:,p)=0;
end
%-------------------------------------------------------------------
%--------------------Vega limit-------------------------------------
CallOptions(:,p)=CallOptions(:,p)...
    .*max(sign(abs(CallOptions(:,p).*ATMCallVega(:,p))-VegaLimit),0);
%-------------------------------------------------------------------
%--------------Updates----------------------------------------------
if p~=1 & mod(p,TradingDayInterval)~=0
    CallOptions(:,p)=0;
end
%-------------------------------------------------------------------
%------------Delta hedge--------------------------------------------
if p==1
    Delta(:,p)=interpn(X,Y,real(FFTDigitalDelta)...
        ,S(:,p),V(:,p),'linear')-CallOptions(:,p).*ATMCallDelta(:,p);
```

```
    elseif p<=FinalTradingDay
        PortfolioDelta(:,p)=sum(CallOptions(:,[1:p]).*...
            interpn(X,Y,real(FFTCallDelta),S(:,p)*ones(1,p).*...
            K./S(:,[1:p]),V(:,p)*ones(1,p),'linear'),2);

        Delta(:,p)=interpn(X,Y,real(FFTDigitalDelta)...
            ,S(:,p),V(:,p),'linear')-PortfolioDelta(:,p);
    else
        Delta(:,p)=Delta(:,FinalTradingDay);
    end
    %-------------------------------------------------------------------
    %-----------------Portfolio value evolution-------------------------
    if p==1
        PortfolioValue(:,p)=-CallOptions(:,p).*ATMCallPrice(:,p)...
            -Delta(:,p).*S(:,p);
    else
        PortfolioValue(:,p)=PortfolioValue(:,p-1).*exp(r*dt)...
            +S(:,p-1).*Delta(:,p-1)*(exp(d*dt)-1)...
            -S(:,p).*(Delta(:,p)-Delta(:,p-1))...
            -CallOptions(:,p).*ATMCallPrice(:,p)...
            -PriceSpread(:,p).*abs(CallOptions(:,p));
    end
end
%--------Portfolio value at maturity time---------------------------------
PortfolioValue(:,timesteps+1)=PortfolioValue(:,timesteps).*exp(r*dt)...
    +S(:,timesteps).*Delta(:,timesteps)*(exp(d*dt)-1)...
    -max(sign(S(:,timesteps+1)-K),0)...
    +sum(CallOptions(:,1:timesteps).*...
    max(S(:,timesteps+1)*ones(1,timesteps)-S(:,1:timesteps),0),2)...
    +Delta(:,timesteps).*S(:,timesteps+1);
%-------------------------------------------------------------------------
%--------------Price------------------------------------------------------
BSDigital=exp(-r)*normcdf(-(r-d-1/2*BSVolatility^2)/BSVolatility)
BatesDigitalPrice=exp(-r)*mean(PortfolioValue(:,timesteps+1))
y='done';


function y=QStar(k,V,x,tau,r,d)
kappa=x(1);
theta=x(2);
epsilon=x(3);
rho=x(4);
```

```
lambda=x(5);
ny=x(6);
delta=x(7);
u=kappa-rho.*epsilon./2;
xi=sqrt(k.^2.*epsilon.^2.*(1-rho.^2)...
    +2.*i.*k.*rho.*epsilon.*u +u.^2+epsilon.^2./4);
PsiPlus=-(u+i.*k.*rho.*epsilon)+xi;
PsiMinus=(u+i.*k.*rho.*epsilon)+xi;
A=-(kappa.*theta./(epsilon.^2)).*...
    (PsiPlus.*tau+2.*log((PsiMinus+PsiPlus.*exp(-xi.*tau))./(2.*xi)));
LAMBDA=exp(-i.*k.*(ny+delta.^2./2)-(k.^2-1./4).*delta.^2./2+ny./2)-...
    1-(-i.*k+1./2).*(exp(ny+delta.^2./2)-1);
D=tau.*LAMBDA;
y=exp(A+B(k,x,tau,r,d).*V+D.*lambda);


function y=B(k,x,tau,r,d)
kappa=x(1);
theta=x(2);
epsilon=x(3);
rho=x(4);
lambda=x(5);
ny=x(6);
delta=x(7);
u=kappa-rho.*epsilon./2;
xi=sqrt(k.^2.*epsilon.^2.*(1-rho.^2)...
    +2.*i.*k.*rho.*epsilon.*u +u.^2+epsilon.^2./4);
PsiPlus=-(u+i.*k.*rho.*epsilon)+xi;
PsiMinus=(u+i.*k.*rho.*epsilon)+xi;
y=-(k.^2+1./4).*(1-exp(-xi.*tau))./(PsiMinus+PsiPlus.*exp(-xi.*tau));


function y=ImpliedVolatilityBisections(S,K,r,d,tau,option_price)
Y1=0;
Y2=0;
Y3=0;
Y4=0;
ACCURACY=1e-6;
MAX_ITERATIONS=200;
HIGH_VALUE=1e10;
ERR=-1e40;
%------arbitrage test-----------------------------------------------------
```

```
sigma_low=0.0001;
price=BlackScholesEuro(S,K,r,d,tau,sigma_low);
if(price>option_price)
    Y1=ERR;
end
%-------------------------------------------------------------------------
sigma_high=0.3;
price=BlackScholesEuro(S,K,r,d,tau,sigma_high);
while(price<option_price)
    sigma_high=2*sigma_high;
    price=BlackScholesEuro(S,K,r,d,tau,sigma_high);
    if(sigma_high>HIGH_VALUE)
        Y2=ERR;
    end
end
for i=0:MAX_ITERATIONS
    sigma=(sigma_low+sigma_high)*0.5;
    price=BlackScholesEuro(S,K,r,d,tau,sigma);
    test=(price-option_price);
    if(abs(test)<ACCURACY)
        Y3=sigma;
        break
    end
    if(test<0)
        sigma_low=sigma;
    else
        sigma_high=sigma;
    end
    if(i==MAX_ITERATIONS)
        Y4=ERR;
    end
end
if (Y1~=0)
    y=Y1;
elseif (Y2~=0)
    y=Y2;
elseif(Y4~=0)
    y=Y4;
else
    y=Y3;
end
```

```
function y=BlackScholesEuro(S,K,r,d,tau,sigma)
dplus=1/sigma/sqrt(tau)*(log(S/K)+(r-d+sigma^2/2)*tau);
dminus=1/sigma/sqrt(tau)*(log(S/K)+(r-d-sigma^2/2)*tau);
y=S*exp(-d*tau)*normcdf(dplus)-K*exp(-r*tau)*normcdf(dminus);
```

References

[**Gla**] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer-Verlag, New York, 2004.

[**Jäc**] P. Jäckel, *Monte Carlo methods in Finance*, Wiley, Chichester, 2002.

[**Sep**] A. Sepp, *Pricing European-Style Options under Jump Diffusion Processes with Stochastic Volatility: Applications of Fourier Transform*, University of Tartu, 2003.

[**Nor**] T. Nordqvist, *TrippleDigital*, Svenska Handelsbanken AB, Stockholm, 2005.

[**Sob**] I.M. Sobol, Uniformly distributed sequences with additional uniform properties, *USSR Computational Mathematics and Mathematical Physics* 16 (5): 238-242, 1967.

[**BS**] F. Black and M. Scholes, The pricing of options and corporate liabilities, *J. Polit. Econ.* 81 (3), 637-659, 1973.

[**AHNW**] E. Ayache, P. Henrotte, S. Nassar and X. Wang, Can anyone solve the smile problem. Preprint. Available at `http://www.ito33.com/html/print`

[**Bat**] D.S. Bates, Jumps and Stochastic Volatility; Exchange Rate Processes Implicit in Deutsche Mark Options, *The Review of Financial Studies* 9 (1): 69-107, 1996.

[**CIR**] J.C. Cox, J.E. Ingersoll and S. Ross, A theory of the term structure of interest rates, *Econometrica* 53 (2): 373-384, 1985.