VIBRATO MONTE CARLO AND THE CALCULATION OF GREEKS

Dr. Sinéad Keegan

The thesis is submitted to
Oxford University
for the degree of
M.Sc. in Mathematical and Computational Finance

June 2008

under the supervision of
Prof. Michael B. Giles

Abstract

In computational finance Monte Carlo simulation can be used to calculate the correct prices of financial options, and to compute the values of the associated Greeks (the derivatives of the option price with respect to certain input parameters). The main methods used for the calculation of Greeks are finite difference, likelihood ratio, and pathwise sensitivity. Each of these has its limitations and in particular the pathwise sensitivity approach may not be used for an option whose payoff function is discontinuous. Vibrato Monte Carlo is a new idea that addresses the limitations of previous methods; it combines the pathwise sensitivity approach for the SDE path calculation with the likelihood ratio method for payoff evaluation. This thesis discusses Vibrato Monte Carlo approximations for a digital option on an asset following one-dimensional geometric Brownian motion.

1 Introduction

1.1 Monte Carlo

Monte Carlo is a technique that can be used to simulate or approximate integrals or expectations that, for one reason or another, cannot be computed analytically. The method involves taking random samples from a collection of all possible outcomes; the fraction of samples that falls in a particular set is then used as an estimate of that set's volume. By the law of large numbers, this estimate converges to the correct value as the number of samples increases. Some useful information about the size of the error in the estimate is given by the central limit theorem.

To demonstrate the concept of Monte Carlo, consider estimating the integral of a function f over the unit interval. The integral

$$\alpha = \int_0^1 f(x)dx$$

can be represented as an expectation

$$\mathbb{E}\left[f(U)\right] \ ,$$

where $U \sim \text{Unif}[0,1]$. To calculate the Monte Carlo estimate of the integral α, M points $U_1, \ldots, U_M$ are chosen independently from Unif$[0,1]$, f is evaluated at each of these points, and the result is averaged to give

$$\hat{\alpha}_M = M^{-1} \sum_{m=1}^{M} f(U_m) \ .$$

By the strong law of large numbers, $\hat{\alpha}_M \to \alpha$ as $M \to \infty$, with probability 1, assuming f is integrable over $[0,1]$. The error in the Monte Carlo estimate is $\hat{\alpha}_M - \alpha \sim \text{Normal}\left(0, \sigma_f/\sqrt{M}\right)$, where σ_f is defined by

$$\sigma_f^2 = \int_0^1 \left(f(x) - \alpha\right)^2 dx \ .$$

The quantity σ_f is generally unknown but can be estimated using the sample standard deviation

$$s_f = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(f(U_m) - \hat{\alpha}_M\right)^2} \ .$$

Monte Carlo techniques are used in many areas of mathematics and science. In particular they are becoming increasingly popular in the field of computational finance. Option prices in finance can often be represented as expected values, making them very suitable for Monte Carlo estimation. Evaluating an option by Monte Carlo typically involves simulating paths of stochastic processes used to describe the underlying asset prices. For a European option depending only on one asset, its price can be expressed as

$$V = \mathbb{E}\left[f\left(S(T)\right)\right] \ ,$$

where S denotes the asset price, T denotes the maturity time, and f is the payoff function of the option. Monte Carlo estimates this expectation by simulating a finite number of future paths and averaging over this finite set:

$$\hat{V} = M^{-1}\sum_{m=1}^{M} f\left(S^{(m)}\right) \ ,$$

where each $S^{(m)}$ is a simulated solution of the asset price SDE.

More importantly, Monte Carlo can be used to estimate the Greeks. The Greeks are the sensitivities of the option value to changes in particular input parameters, and are important in hedging and in risk analysis. The most

commonly used Greeks are

$$\begin{aligned} \Delta &= \frac{\partial V}{\partial S_0} , \\ \Gamma &= \frac{\partial \Delta}{\partial S_0} , \\ \rho &= \frac{\partial V}{\partial r} , \\ \text{Vega} &= \frac{\partial V}{\partial \sigma} . \end{aligned}$$

Option prices are to a certain extent determined by the market (since models are usually calibrated to match market values), however simulation is the only way to compute the Greeks. The three major approaches used in the Monte Carlo simulation of Greeks are finite difference, the likelihood ratio method, and the pathwise sensitivity approach.

Monte Carlo has many advantages; it is a simple, flexible technique, and can be parallelised easily. For very small dimensions it is less efficient than finite difference methods, however it is extremely good at dealing with high-dimensional problems. Monte Carlo is not yet efficient for applications with optional exercise but this is an active area of research.

1.2 Stochastic Differential Equations

A stochastic differential equation has the general form

$$dS(t) = a(S,t)\,dt + b(S,t)\,dW(t) , \tag{1}$$

where $a(S,t)$ denotes the drift, $b(S,t)$ denotes the volatility, and $W(t)$ is a standard Brownian motion. The stochastic term $b(S,t)\,dW(t)$ models the unpredictable events that influence the asset price.

In the simplest cases equation (1) can be integrated, allowing $S(T)$ to be

calculated explicitly. For example scalar geometric Brownian Motion

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

can be integrated to get

$$S(T) = S(0)\exp\left\{\left(r - \frac{1}{2}\sigma^2\right)T + \sigma W(T)\right\}\ .$$

Here the risk-free interest rate r and the volatility σ are both constant.

In most cases, exact integration of equation (1) is impossible. Instead, some discretisation of the SDE is used to numerically calculate $S(T)$. The Euler-Maruyama method is the simplest such discretisation.

1.2.1 The Euler-Maruyama Discretisation

The Euler-Maruyama method is used if either

- the SDE cannot be integrated exactly, or
- the SDE is integrable but the payoff is path dependent.

To implement the Euler-Maruyama method, the time interval $[0, T]$ is split into N timesteps of equal length $h = T/N$. The Euler-Maruyama discretisation of the SDE

$$dS(t) = a(S,t)\,dt + b(S,t)\,dW(t)\ ,$$

is given by

$$\hat{S}_{n+1} = \hat{S}_n + a(\hat{S}_n, t_n)h + b(\hat{S}_n, t_n)\Delta W_n\ ,$$

where h is the timestep length, $\hat{S}_n$ is the approximation to $S(nh)$, and ΔW_n are independent identically distributed Normal$(0, h)$ Brownian increments. ΔW_n can be defined by $\Delta W_n = LZ_n$, where elements of Z_n are independent identically distributed Normal$(0, 1)$ random variables, and $LL^T = \mathrm{Cov}(\Delta W_n)$, where Cov denotes the covariance matrix. In an Euler-Maruyama discretisation, the production of each output $\hat{S}_N$ involves N random inputs W_n. To calculate the Monte Carlo estimate $\hat{V}$, where the asset price SDE

has been discretised using the Euler-Maruyama method, $S^{(m)}$ is replaced by the approximation $\hat{S}_N^{(m)}$, the value at the end of the N^{th} timestep in the discretisation of the SDE.

Once the asset price has been determined, the option price and corresponding Greeks can be computed. The following sections briefly describe the three most common methods for calculating Greeks via Monte Carlo simulation. A more in depth discussion of these topics can be found in Glasserman's book [1].

1.3 Monte Carlo Calculation of Greeks

1.3.1 Finite Difference Method

Consider the option value as a function of θ,

$$V(\theta) = \mathbb{E}\left[f(S(T))\right] \ ,$$

where θ is some input parameter and V is sufficiently differentiable with respect to θ. The first and second order derivatives of V with respect to θ can be calculated by

$$\begin{aligned} \frac{\partial V}{\partial \theta} &\approx \frac{V(\theta + \Delta\theta) - V(\theta - \Delta\theta)}{2\Delta\theta} \ , \\ \frac{\partial^2 V}{\partial \theta^2} &\approx \frac{V(\theta + \Delta\theta) - 2V(\theta) + V(\theta - \Delta\theta)}{(\Delta\theta)^2} \ . \end{aligned}$$

The finite difference method has the advantage of being simple and easy to implement. On the downside it is computationally expensive, and requires a careful choice of $\Delta\theta$ for each input parameter θ. $\Delta\theta$ too small can lead to large variance if the payoff function is not differentiable, while $\Delta\theta$ too large can give rise to significant approximation error, see [1] page 379 for more details.

1.3.2 Likelihood Ratio Method

For a scalar SDE whose terminal probability distribution is known, the option value can be written as

$$V = \mathbb{E}\left[f(S(T))\right] = \int f(S)p(S)\,dS \ ,$$

where $p(S)$ denotes the probability density function of the final state $S(T)$. Any dependence on the input parameters comes entirely through $p(S)$, and not from the payoff function $f(S)$. Hence the first order Greeks are given by

$$\frac{\partial V}{\partial \theta} = \int f\,\frac{\partial p}{\partial \theta}\,dS = \int f\,\frac{\partial(\log p)}{\partial \theta}\,p\,dS = \mathbb{E}\left[f\,\frac{\partial(\log p)}{\partial \theta}\right] \ .$$

The main advantage of the likelihood ratio method is that it does not require a differentiable payoff function, and hence may be used in cases where an option has discontinuous payoff.

For an SDE path simulation with N timesteps, and an option whose payoff is a function purely of the discrete states $\hat{S}_n$, the option value can be written as

$$V = \mathbb{E}\left[f(\hat{S})\right] = \int f(\hat{S})\,p(\hat{S})\,d\hat{S} \ ,$$

where $d\hat{S} = d\hat{S}_1 \ldots d\hat{S}_N$, and $p(\hat{S}) = \prod_n p_n\left(\hat{S}_{n+1}|\hat{S}_n\right)$.

The biggest drawback of the likelihood ratio method is that it leads to a Monte Carlo estimator with variance of order h^{-1}, see [1] page 409. This becomes infinite as $h \to 0$, making the method unsuitable for calculations involving path-dependent options.

1.3.3 Pathwise Sensitivity Method

By a change of variables, the option value can be expressed in the form

$$V = \mathbb{E}\left[f\left(S(T)\right)\right] = \int f\left(S(T)\right)\,p_W(W)\,dW \ ,$$

where $p(W)$ is the probability density function of the driving Brownian motion $W(T)$. Since the θ-dependence arises through f rather that p, the first order Greeks are given by

$$\frac{\partial V}{\partial \theta} = \int \frac{\partial f}{\partial S} \frac{\partial S(T)}{\partial \theta} p_W(W)\, dW = \mathbb{E}\left[\frac{\partial f}{\partial S} \frac{\partial S(T)}{\partial \theta}\right] , \tag{2}$$

where $\partial S(T)/\partial\theta$ is evaluated at fixed W. This leads to the Monte Carlo estimator

$$\frac{\partial \hat{V}}{\partial \theta} = M^{-1} \sum_{m=1}^{M} \frac{\partial f}{\partial S}\left(S^{(m)}\right) \frac{\partial S^{(m)}}{\partial \theta}$$

for $\partial V/\partial\theta$, which is the derivative of the usual price estimator

$$\hat{V} = M^{-1} \sum_{m=1}^{M} f\left(S^{(m)}\right) .$$

Instead of (2), it is sometimes useful to consider an integral with respect to a random variable $Z \sim \text{Normal}(0,1)$, as this allows for calculation of the sensitivity with respect to changes in correlation. Writing

$$V = \mathbb{E}\left[f\left(S(T)\right)\right] = \int f\left(S(T)\right)\, \phi(Z)\, dZ ,$$

where ϕ is the unit normal probability density function, it follows that

$$\frac{\partial V}{\partial \theta} = \mathbb{E}\left[\frac{\partial f}{\partial S} \frac{\partial S(T)}{\partial \theta}\right] ,$$

where $\partial S(T)/\partial\theta$ is evaluated at fixed Z.

The pathwise sensitivity method extends naturally to an SDE path simulation with N timesteps. The option value is written as

$$V = \mathbb{E}\left[f(\hat{S})\right] = \int f(\hat{S}(Z))\phi(Z)dZ ,$$

where $dZ = dZ_1 \dots dZ_N$, and $\phi(Z) = \prod_n \ \phi(Z_n)$. Then

$$\frac{\partial V}{\partial \theta} = \mathbb{E}\left[\frac{\partial f}{\partial \hat{S}}\frac{\partial \hat{S}}{\partial \theta}\right] \ .$$

A huge disadvantage of the pathwise sensitivity method is that the payoff function $f(S)$ needs to be differentiable. One way of getting around this problem is to use "payoff smoothing"; this involves using a continuous piecewise linear function to approximate the payoff function. One possibility is to smooth discontinuities using conditional expectations, as discussed in the next section.

Unlike the likelihood ratio method, the pathwise sensitivity approach does not suffer from the problem of variance "blow-up" as the number of timesteps increases, making it more suitable for dealing with path-dependent options.

The pathwise sensitivity method has a very efficient adjoint implementation [2] whose computational cost is independent of the number of first derivatives to be calculated. This is particularly useful for calculating multiple Greeks simultaneously.

Despite its many benefits, this approach is not perfect. Even when the payoff function is differentiable, the use of scripting in real-world implementations means that it can be difficult in practice to calculate the derivatives of very complicated financial options.

1.4 Technique of Conditional Expectation

Section 7.2.3 of Glasserman's book [1] demonstrates the technique of conditional expectation. In his example a path simulation for a particular path $W(t)$ is carried out in the usual way for the first $N-1$ timesteps. On the final timestep one considers the full distribution of possible values for W.

This gives rise to a Gaussian distribution for $\hat{S}_N$ at maturity time T,

$$p_S\left(\hat{S}_N\right) = \frac{1}{\sqrt{2\pi}\sigma_W} \exp\left(-\frac{\left(\hat{S}_N - \mu_W\right)^2}{2\sigma_W^2}\right) \ .$$

Here $\mu_W = \hat{S}_{N-1} + a(\hat{S}_{N-1}, T-h)h$ and $\sigma_W = b(\hat{S}_{N-1}, T-h)\sqrt{h}$, where $a(S,t)$ and $b(S,t)$ are the drift and volatility of the underlying process (1).

Consider for example a digital option with strike K. Its payoff is

$$f(S(T)) = H(S(T)-K) \equiv \begin{cases} 1, & S(T) > K \\ 0, & S(T) \leq K \ . \end{cases}$$

The conditional expectation for the value of the option is

$$\mathbb{E}\left[f\left(\hat{S}_N\right)|\hat{S}_{N-1}\right] = \int_{-\infty}^{\infty} H(\hat{S}_N - K) p_S(\hat{S}_N) d\hat{S}_N = \Phi\left(\frac{\mu_W - K}{\sigma_W}\right) \ ,$$

where Φ denotes the cumulative Normal distribution function.

The Monte Carlo estimator for the option value is

$$\hat{V} = M^{-1} \sum_m \mathbb{E}\left[f\left(\hat{S}_N\right)|\hat{S}_{N-1}^{(m)}\right] \ .$$

The crucial point is that, since $\mathbb{E}\left[f\left(\hat{S}_N\right)|\hat{S}_{N-1}\right]$ is differentiable in the input parameters, the pathwise sensitivity approach can now be applied.

There are two problems with this technique. Firstly the conditional expectation often gives rise to a multi-dimensional integral that can't be solved easily. Secondly it requires a change to the complex software frameworks used in the finance industry to specify payoffs. The next section discusses a technique that addresses both of these issues.

1.5 Vibrato Monte Carlo

Vibrato Monte Carlo is a new approach to the calculation of Greeks. It addresses many of the limitations of previously described methods by combining the (adjoint) pathwise sensitivity approach for stochastic path evolution with the likelihood ratio method for payoff evaluation. Vibrato Monte Carlo builds on Glasserman's technique of conditional expectation. Following on from this idea, each path simulation for a particular discrete set of increments W computes a Gaussian probability distribution $p_S(\hat{S}_N|W)$ for the state of the path approximation at time T. For a scalar SDE, if μ_W and σ_W are the mean and standard deviation for a given W, then

$$\hat{S}_N(W, Z) = \mu_W + \sigma_W Z \ ,$$

where $Z \sim \text{Normal}(0, 1)$. The expected payoff is

$$\begin{aligned} V &= \mathbb{E}_W \left[\mathbb{E}_Z \left[f(\hat{S}_N)|W \right] \right] \\ &= \int \left\{ \int f(\hat{S}_N) p_S(\hat{S}_N|W) \, d\hat{S}_N \right\} p_W(W) \, dW \ . \end{aligned} \tag{3}$$

The sensitivity of V with respect to an input parameter θ is calculated in two steps. Firstly the pathwise sensitivity approach is applied at fixed W to compute $\partial\mu_W/\partial\theta$ and $\partial\sigma_W/\partial\theta$. Next the likelihood ratio method is applied to the inner conditional expectation in (3) to get

$$\frac{\partial V}{\partial \theta} = \mathbb{E}_W \left[\frac{\partial}{\partial \theta} \mathbb{E}_Z \left[f(\hat{S}_N)|W \right] \right] = \mathbb{E}_W \left[\mathbb{E}_Z \left[f(\hat{S}_N) \frac{\partial(\log p_S)}{\partial \theta} |W \right] \right] \ .$$

The following equations hold:

$$p_S(\hat{S}_N) = \frac{1}{\sqrt{2\pi}\sigma_W} \exp \left\{ -\frac{(\hat{S}_N - \mu_W)^2}{2\sigma_W^2} \right\} \ , \tag{4}$$

$$\frac{\partial(\log p_S)}{\partial \theta} = \frac{\partial(\log p_S)}{\partial \mu_W} \frac{\partial \mu_W}{\partial \theta} + \frac{\partial(\log p_S)}{\partial \sigma_W} \frac{\partial \sigma_W}{\partial \theta} \ , \tag{5}$$

with μ_W and σ_W as before.

The Monte Carlo estimators for V and $\partial V/\partial\theta$ are

$$\hat{V} = M^{-1}\sum_m \tilde{\mathbb{E}}_Z\left[f\left(\hat{S}_N\right)|W\right], \qquad \frac{\partial\hat{V}}{\partial\theta} = M^{-1}\sum_m \tilde{\mathbb{E}}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\theta}|W\right],$$

where $\tilde{\mathbb{E}}$ denotes a conditional expectation estimate. It is important that these estimates are as efficient as possible. These low variance estimates are constructed as follows [2].

The symmetry of the probability distribution of Z allows one to use the identity $\mathbb{E}_Z[h(Z)] = \mathbb{E}_Z[h(-Z)]$ for all functions $h(Z)$. Then for a given W

$$\begin{aligned}\tilde{\mathbb{E}}_Z\left[f\left(\hat{S}_N\right)\right] &= f(\mu_W) + \mathbb{E}_Z\left[f(\mu_W+\sigma_W Z) - f(\mu_W)\right] \\ &= f(\mu_W) + \mathbb{E}_Z\left[\frac{1}{2}\left(f\left(\mu_W+\sigma_W Z\right) - 2f(\mu_W) + f(\mu_W-\sigma_W Z)\right)\right].\end{aligned}$$

For differentiable $f(S)$, the second term is the expectation of a quantity with very small mean and variance, so a single sample should be sufficient for the estimator.

For a one-dimensional SDE, it follows from equations (4) and (5) that

$$\begin{aligned}\mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\theta}\right] &= \frac{\partial\mu_W}{\partial\theta}\mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\mu_W}\right] \\ &+ \frac{\partial\sigma_W}{\partial\theta}\mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\sigma_W}\right]. \qquad (6)\end{aligned}$$

Looking at both of these expectations separately, the first one can be written

as

$$\begin{aligned}
\tilde{\mathbb{E}}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\mu_W}\right] &= \mathbb{E}_Z\left[\frac{\hat{S}_N-\mu_W}{\sigma_W^2}f\left(\hat{S}_N\right)\right] \\
&= \mathbb{E}_Z\left[\frac{Z}{\sigma_W}f\left(\mu_W+\sigma_W Z\right)\right] \\
&= \mathbb{E}_Z\left[\frac{Z}{2\sigma_W}\left(f\left(\mu_W+\sigma_W Z\right)-f\left(\mu_W-\sigma_W Z\right)\right)\right] .
\end{aligned}$$

For $f(S)$ differentiable, this is the expectation of a quantity with $\mathcal{O}(1)$ magnitude. One sample should be sufficient, but it may be better to use a few samples if the computational cost of evaluating the payoff is small compared to the path calculation. If $f(S)$ is discontinuous then the expectation is an $\mathcal{O}(h^{-1/2})$ quantity for paths near the discontinuity, and in this case several paths should definitely used to estimate the expected value.

Using the additional result that $\mathbb{E}_Z\left[Z^2-1\right]=0$, the second expectation in equation (6) may be written as

$$\begin{aligned}
\tilde{\mathbb{E}}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\sigma_W}\right] &= \mathbb{E}_Z\left[\left(-\frac{1}{\sigma_W}+\frac{\left(\hat{S}_N-\mu_W\right)^2}{\sigma_W^3}\right)f\left(\hat{S}_N\right)\right] \\
&= \mathbb{E}_Z\left[\frac{Z^2-1}{\sigma_W}f\left(\mu_W+\sigma_W Z\right)\right] \\
&= \mathbb{E}_Z\left[\frac{Z^2-1}{\sigma_W}\left(f\left(\mu_W+\sigma_W Z\right)-f(\mu_W)\right)\right] \\
&= \mathbb{E}_Z\left[\frac{Z^2-1}{2\sigma_W}\left(f\left(\mu_W+\sigma_W Z\right)-2f(\mu_W)+f\left(\mu_W-\sigma_W Z\right)\right)\right] .
\end{aligned}$$

In general this quantity is no larger than the previous expectation, therefore the same set of samples will suffice.

1.6 Multi-dimensional SDEs

Vibrato Monte Carlo can be extended to the case of multiple assets with a multivariate Gaussian distribution conditional on the discrete set of Brownian increments. If μ_W is the column vector of means and Σ_W is the covariance matrix, then $\hat{S}_N$ can be written as

$$\hat{S}_N(W, Z) = \mu_W + CZ \ ,$$

where Z is a vector of uncorrelated Normal$(0, 1)$ random variables and C is any matrix such that $\Sigma_W = CC^T$.

The joint probability density function for S is given by

$$\log p_S = -\frac{1}{2}\log|\Sigma| - \frac{1}{2}\left(\hat{S}_N - \mu_W\right)^T \Sigma^{-1}\left(\hat{S}_N - \mu_W\right) - \frac{1}{2}d\log(2\pi) \ ,$$

where d is the dimension of the problem. It is easily shown that

$$\begin{aligned}
\frac{\partial \log p_S}{\partial \mu_W} &= C^{-T}Z \ , \\
\frac{\partial \log p_S}{\partial \Sigma_W} &= \frac{1}{2}C^{-T}\left(ZZ^T - I\right)C^{-1} \ .
\end{aligned}$$

Then for a given W

$$\begin{aligned}
\mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log\, p_S)}{\partial\theta}\right] &= \left(\frac{\partial\mu_W}{\partial\theta}\right)^T \mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log\, p_S)}{\partial\mu_W}\right] \\
&+ \ \text{Trace}\left(\frac{\partial\Sigma_W}{\partial\theta}\,\mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log\, p_S)}{\partial\Sigma_W}\right]\right) \ .
\end{aligned}$$

Efficient estimators are given by

$$\mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\mu_W}\right] = \mathbb{E}_Z\left[\frac{1}{2}\left(f\left(\mu_W + CZ\right) - f\left(\mu_W - CZ\right)\right)C^{-T}Z\right]$$

and

$$\begin{aligned} & \mathbb{E}_Z\left[f\left(\hat{S}_N\right)\frac{\partial(\log p_S)}{\partial\Sigma_W}\right] \\ = \; & \mathbb{E}_Z\left[\frac{1}{4}\left(f\left(\mu_W+CZ\right)-2f(\mu_W)+f\left(\mu_W-CZ\right)\right)C^{-T}(ZZ^T-I)C^{-1}\right] . \end{aligned}$$

2 Simulations and Results

2.1 Objectives

- Show that vibrato Monte Carlo approximations are close to analytic values.
- Investigate the optimal number of samples.
- Investigate the benefits of using the lower variance conditional expectation estimates in vibrato calculations.
- Investigate the behaviour of vibrato Monte Carlo as $h \to 0$.

2.2 Vibrato Monte Carlo for 1-dimensional geometric Brownian motion

For all calculations we consider a digital call option on an asset whose price follows one-dimensional geometric Brownian motion. The parameters used

are

$$\begin{aligned} r &= 0.05 \quad \text{(risk-free interest rate)} \\ \sigma &= 0.2 \quad \text{(volatility)} \\ T &= 1 \quad \text{(time interval)} \\ K &= 100 \quad \text{(strike price)} \\ S_0 &= 100 \quad \text{(initial asset value)} \\ N &= 64 \quad \text{(number of timesteps)} \\ M &= 100000 \quad \text{(number of paths)} \ . \end{aligned}$$

For the value, delta, and vega of the option we compute the analytic value [5] and also the vibrato Monte Carlo approximation (with corresponding confidence interval). The vibrato calculation is done twice, once using low variance estimators, and once with ordinary estimators. In each case vibrato Monte Carlo using low variance estimators yields an approximation that is very close to the corresponding analytic value. Calculations are carried out using the Matlab files digital_call.m [4], estimator_comparison.m, and vibratos.m. The results are displayed below.

```
analytic value: 0.532325
vibrato value using lower variance estimator: 0.533345 +/- 0.004518
vibrato value without lower variance estimator: 0.533801 +/- 0.004708
analytic delta: 0.018762
vibrato delta using lower variance estimator: 0.018653 +/- 0.000729
vibrato delta without lower variance estimator: 0.017333 +/- 0.002855
analytic vega: -0.656671
vibrato vega using lower variance estimator: -0.654102 +/- 0.026784
vibrato vega without lower variance estimator: -0.591695 +/- 0.236737
```

The Matlab file LRM.m calculates the same Greeks, this time using the likelihood ratio method. For comparison, the results are given below.

```
Analytic Delta:  0.018762
LRM Delta: 0.018776  +/- 0.000265
Analytic Value:  -0.656671
LRM Vega: -0.650096  +/- 0.042812
```

2.3 Optimal number of samples

Theorem (Giles [3]) If W and Z are independent random variables, then

$$\hat{Y}_{M,d} = M^{-1} \sum_{m=1}^{M} \left(d^{-1} \sum_{i=1}^{d} f(W^{(m)}, Z^{(m,i)}) \right) \ ,$$

with independent samples $W^{(m)}$ and $Z^{(m,i)}$, is an unbiased estimator for

$$\mathbb{E}_{W,Z}\left[f(W,Z)\right] \equiv \mathbb{E}_W\left[\mathbb{E}_Z\left[f(W,Z)\right]\right] \ ,$$

and its variance is

$$\mathbb{V}\left[\hat{Y}_{M,d}\right] = M^{-1}\mathbb{V}_W\left[\mathbb{E}_Z\left[f(W,Z)\right]\right] + (Md)^{-1}\mathbb{E}_W\left[\mathbb{V}_Z\left[f(W,Z)\right]\right] \ . \qquad (7)$$

Here d denotes the number of samples and M is the number of Monte Carlo paths.

One would like to know how the variance of the vibrato Monte Carlo estimator depends on the number of samples used. We use the Matlab file vdcZ.m to calculate the quantity on the right hand side of equation (7), i.e. the variance of the Monte Carlo estimator, for different values of d. The file vdcZcompare.m plots the variance of the estimator against the number of samples (d). The calculations are carried out both for delta and vega, and the results are seen in the two graphs below.

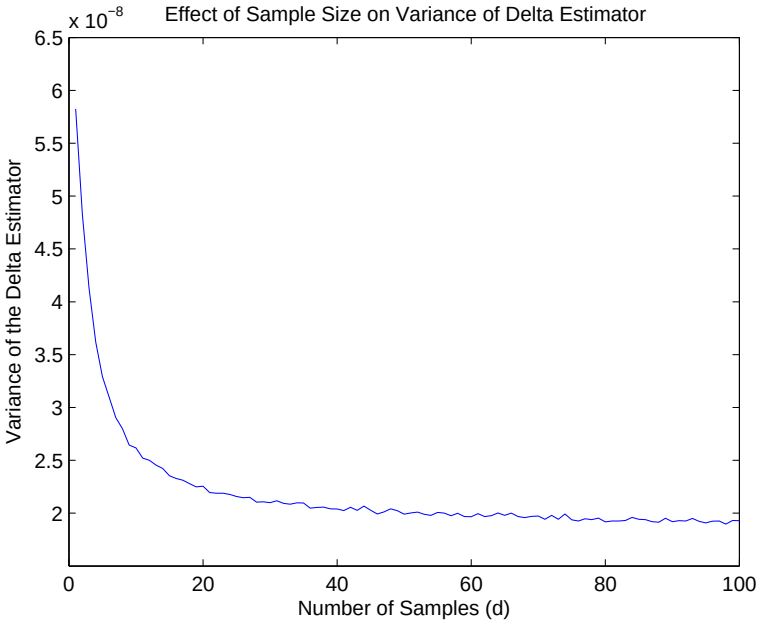


Figure 1: Variance of the Delta Estimator versus Sample Size

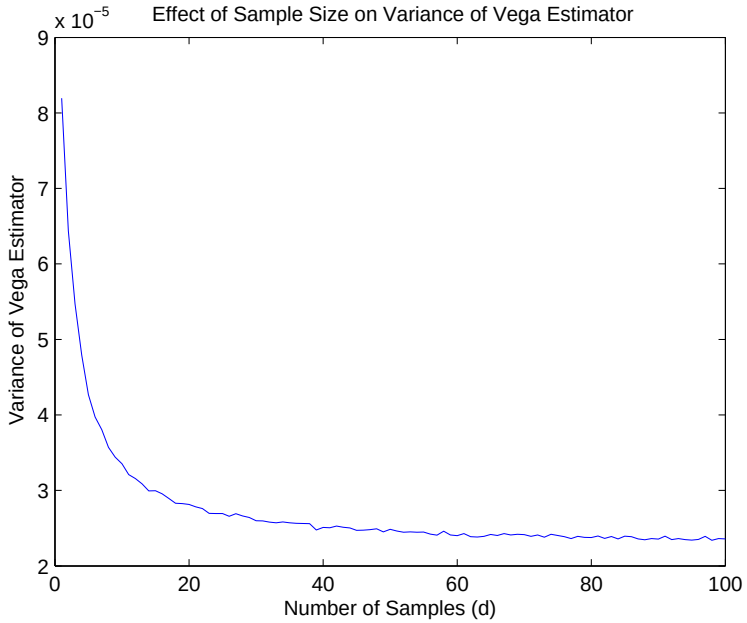


Figure 2: Variance of the Vega Estimator versus Sample Size

2.4 Benefits of low variance estimates

To calculate the value of an option using Vibrato Monte Carlo, we must estimate the expectation

$$\mathbb{E}_{\mathbb{Z}}\left[f(\hat{S}_N)\right] = f(\mu_W) + \mathbb{E}_{\mathbb{Z}}\left[\frac{1}{2}\left(f(\mu_W + \sigma_W Z) - 2f(\mu_W) + f(\mu_W - \sigma_W Z)\right)\right] .$$

The delta and vega of the option involve estimating the quantity

$$\begin{aligned}
& \mathbb{E}_{\mathbb{Z}}\left[f(\hat{S}_N)\frac{\partial(\log p_S)}{\partial\theta}\right] \\
= \; & \frac{\partial\mu_W}{\partial\theta}\mathbb{E}_{\mathbb{Z}}\left[f(\hat{S}_N)\frac{\partial(\log p_S)}{\partial\mu_W}\right] + \frac{\partial\sigma_W}{\partial\theta}\mathbb{E}_{\mathbb{Z}}\left[f(\hat{S}_N)\frac{\partial(\log p_S)}{\partial\sigma_W}\right] \\
= \; & \frac{\partial\mu_W}{\partial\theta}\mathbb{E}_{\mathbb{Z}}\left[\frac{Z}{2\sigma_W}\left(f(\mu_W + \sigma_W Z) - f(\mu_W - \sigma_W Z)\right)\right] \\
+ \; & \frac{\partial\sigma_W}{\partial\theta}\mathbb{E}_{\mathbb{Z}}\left[\frac{Z^2-1}{2\sigma_W}\left(f(\mu_W + \sigma_W Z) - 2f(\mu_W) + f(\mu_W - \sigma_W Z)\right)\right] .
\end{aligned}$$

The right hand side of each of these equations uses a low variance version of the necessary estimator. We investigate the benefits of using the lower variance estimates rather than the ordinary estimates. Our results show that the quantities calculated using vibrato Monte Carlo with low variance estimates, give a more accurate approximation of the analytic value, than vibrato results got using ordinary estimators. This is particularly true as the timestep size $h \to 0$.

2.5 Behaviour as $h \to 0$

We are interested to know how vibrato Monte Carlo approximations are affected by decreasing timestep size. We study this by considering the limit $h \to 0$ (or alternatively $N \to \infty$). Of particular interest is whether there is a variance "blow-up" as $h \to 0$, as is seen with the likelihood ratio method. The Matlab file timestep_size.m compares the analytic value/delta/vega to the corresponding vibrato quantities and plots the results for $N = 1, 2, \ldots, 100$.

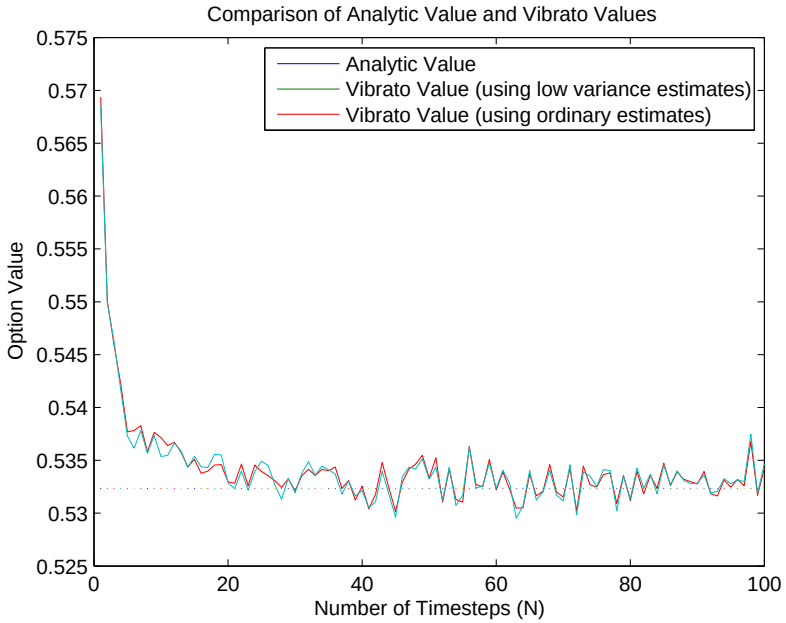


Figure 3: Analytic Value and Vibrato Monte Carlo Estimates

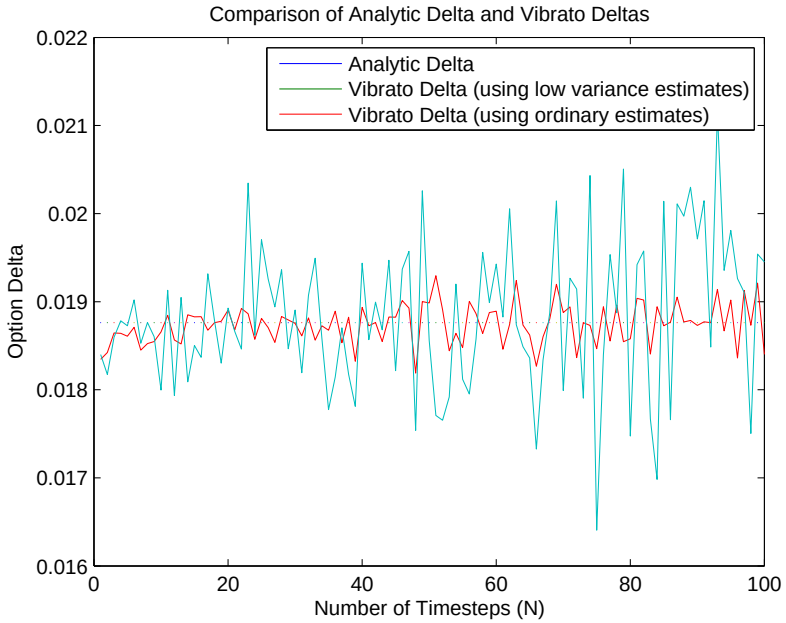


Figure 4: Analytic Delta and Vibrato Monte Carlo Estimates

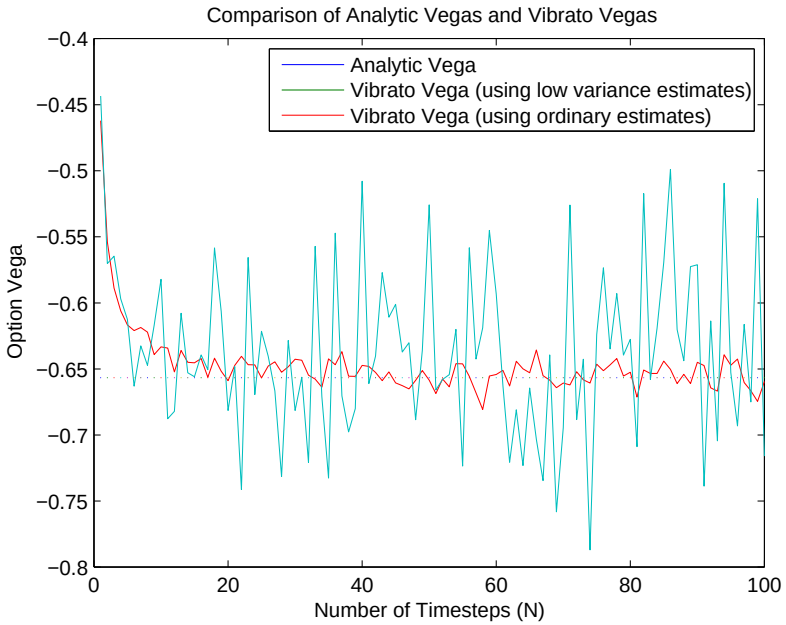


Figure 5: Analytic Vega and Vibrato Monte Carlo Estimates

One expects that the low variance estimator should have variance proportional to $h^{-\frac{1}{2}}$, while the standard estimator should have variance proportional to h^{-1}; for more details see [3]. The following two graphs plot, for delta and vega respectively, the variance of the Vibrato Monte Carlo approximation using low variance estimates and the variance of the approximation using ordinary estimates. In both cases the use of low variance estimates significantly reduces variance.

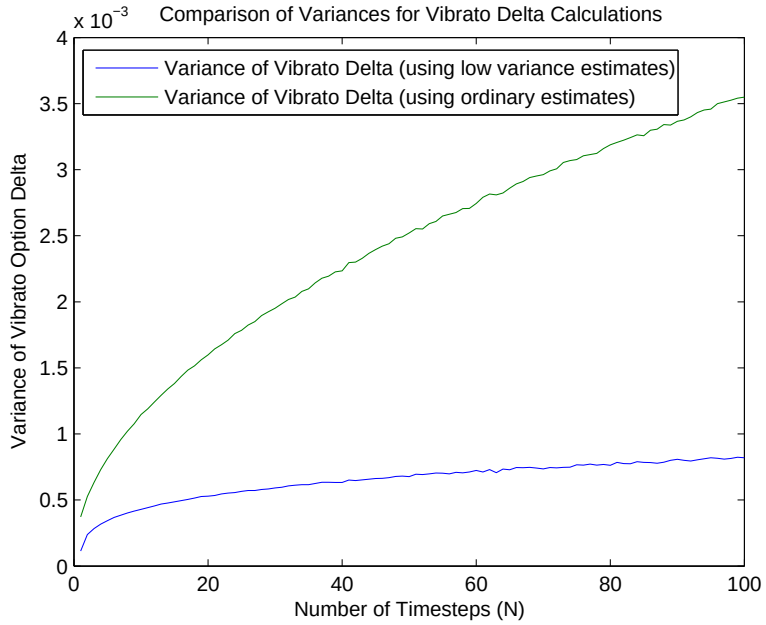


Figure 6: Variances of the Vibrato Monte Carlo Approximations for Delta

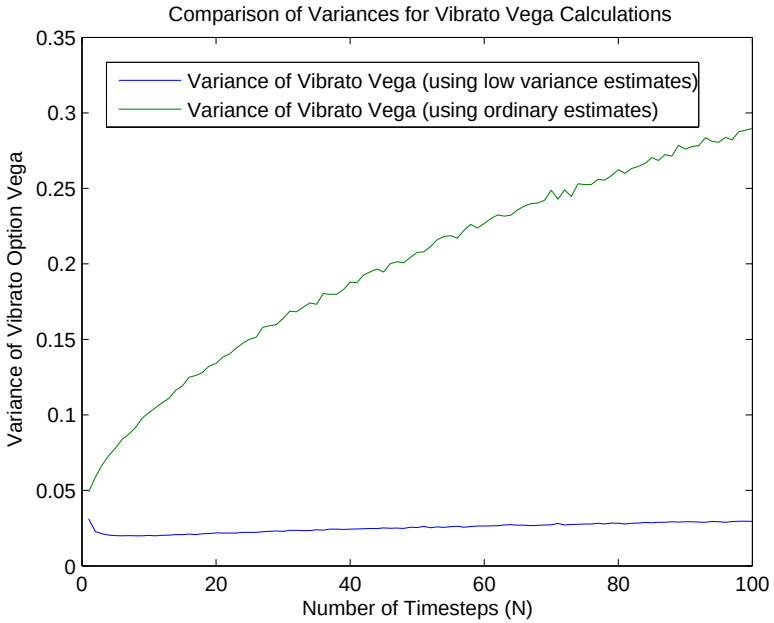


Figure 7: Variances of the Vibrato Monte Carlo Approximations for Vega

3 Conclusions

In this thesis we have described the Vibrato Monte Carlo technique for the evaluation of financial options and their Greeks. This is a hybrid method that combines the pathwise sensitivity approach to path simulation with the likelihood ratio method for payoff evaluation. Vibrato Monte Carlo inherits the computational efficiency of the pathwise method, along with the ease-of-implementation of the likelihood ratio method. More importantly, it can be applied to options with discontinuous payoff functions. We have seen that, for a digital option on a one-dimensional geometric Brownian motion, Vibrato Monte Carlo accurately approximates the correct analytic values of both the price of the option and the Greeks. We have also discussed the effects of timestep size in the path simulation, the optimal number of samples to use in the Monte Carlo calculation, and the benefits of using lower variance estimates for the expected values. Vibrato Monte Carlo should extend easily to a multi-dimensional setting and an obvious future research topic would be to examine this in more detail.

References

[1] Paul Glasserman: *Monte Carlo Methods in Financial Engineering*

[2] Michael B. Giles: *Monte Carlo evaluation of sensitivities in computational finance*

[3] Michael B. Giles: *"Vibrato" Monte Carlo and hybrid Greeks*

[4] M. Giles: *"M.Sc. Course Materials"*, 2008.

[5] Paul Wilmott, Sam Howison, Jeff Dewynne: *The Mathematics of Financial Derivatives.*

A Matlab Code

Filename: "vdcZ.m"

This code calculates the variance of the vibrato Monte Carlo (low variance) estimators used to calculate delta and vega.

```
% function V = vdcZ(r, sigma, T, S0, K, M, N, d, opt)
%
% European digital call with asset price satisfying 1-dimensional geometric
% Brownian motion (Euler-Maruyama approx). vdcZ checks whether the number
% of samples Z, and paths W chosen, satisfies the equation for
% expectation/variance given in the appendix of Mike Giles' paper (by
% calculating the quantity on the RHS of the equation).
%
% r     - interest rate
% sigma - volatility
% T     - time interval
% S0    - initial asset value
% K     - strike price
% opt   - 'delta' or 'vega'
% M     - number of Monte Carlo paths (no. of W's)
% N     - number of timesteps
% d     - number of samples (no. of Z's)
% E     - expectation over W, of the variance over Z, of f(W,Z)
% V     - variance over W, of the expectation over Z, of f(W,Z)
%
% Note: the M and d we use here correspond to the M and N used by Mike
% Giles in his paper.

function V = vdcZ(r, sigma, T, S0, K, M, N, d, opt)
if nargin ~= 9
    error('wrong number of arguments');
end
```

```
h = T/N;
S = S0*ones(1,M);
dS = ones(1,M);    % Used in delta calculation
ds = zeros(1,M);   % Used in vega calculation
    for n = 1:N-1
        dW = sqrt(h)*randn(1,M);
        ds = ds.*(1 + r*h + sigma*dW) + dW.*S;   % For vega
        S = S.*(1 + r*h + sigma*dW);
        dS = dS.*(1 + r*h + sigma*dW);           % For delta
    end
muw = S + r*S*h;
sigw = sigma*S*sqrt(h);
dmuw = (1+r*h)*dS;            % For delta
dsigw = sigma*sqrt(h)*dS;     % For delta
dmuws = (1+r*h)*ds;                        % For vega
dsigws = sigma*sqrt(h)*ds + sqrt(h)*S;    % For vega
Z = randn(d,M);
SN = ones(d,1)*muw + Z.*(ones(d,1)*sigw);
SNm = ones(d,1)*muw - Z.*(ones(d,1)*sigw);
fSN = (0.5)*(1 + sign(SN - K));
fSNm = (0.5)*(1 + sign(SNm - K));
fmuw = ones(d,1)*(0.5)*(1 + sign(muw - K));
invsigw = sigw.^(-1);
e1 = (0.5*Z.*(ones(d,1)*invsigw)).*(fSN-fSNm);                % For delta
e2 = (0.5*(Z.^2 - 1).*(ones(d,1)*invsigw)).*(fSN-2*fmuw+fSNm);  % For vega
% Functions that correspond to the f in Mike Giles' paper (for delta and
% vega respectively).
f = (ones(d,1)*dmuw).*e1 + (ones(d,1)*dsigw).*e2;
% f is the f of Mike Giles' paper for Delta
F = (ones(d,1)*dmuws).*e1 + (ones(d,1)*dsigws).*e2;
% F is the f of Mike Giles' paper for Vega
% We define the following quantities to make the calculation of expectation
% and variance less awkward
```

```
f2 = f.^2; % matrix of f^2
F2 = F.^2; % matrix of F^2
g = sum(f)/d; % Expectation wrt Z of f
G = sum(F)/d; % Expectation wrt Z of F
g2 = sum(f2)/d; % Expectation wrt Z of f^2
G2 = sum(F2)/d; % Expectation wrt Z of F^2
%Expectation over W of the variance over Z of f(W,Z)
EwVzfd = sum((g2-g.^2))/M; % For delta
EwVzfv = sum((G2-G.^2))/M;   % For vega
% Variance over W of the expectation over Z of f(W,Z)
VwEzfd = sum((g.^2))/M - (sum(g)/M)^2; % For delta
VwEzfv = sum((G.^2))/M - (sum(G)/M)^2;   % For vega
switch opt
    case 'delta'
        V = (1/M)*VwEzfd +(1/(M*d))*EwVzfd;
    case 'vega'
        V = (1/M)*VwEzfv +(1/(M*d))*EwVzfv;
    otherwise
        error('invalid value for opt; must be ''delta'' or ''vega''');
end
```

Filename: "vdcZcompare.m"

This code compares the variances calculated by vdcZ.m for different sample sizes (different numbers of Z's).

```
% Looks at how the variance of the estimator (for either delta of vega)
% depends on the number of samples (no. of Z's) used.
r = 0.05;
sigma = 0.2;
T = 1;
S0 = 100;
K = 100;
```

```
N = 64;
M = 100000;
h = T/N;
var_delta = 0;
var_vega = 0;
for d = 1:1:100
    d;
    var_delta(d) = vdcZ(r, sigma, T, S0, K, M, N, d, 'delta');
    var_vega(d) = vdcZ(r, sigma, T, S0, K, M, N, d, 'vega');
end
d = 1:1:100;
plot(d,var_delta)
title 'Effect of Sample Size on Variance of Delta Estimator'
xlabel 'Number of Samples (d)'
ylabel 'Variance of the Delta Estimator'
d = 1:1:100;
plot(d,var_vega)
title 'Effect of Sample Size on Variance of Vega Estimator'
xlabel 'Number of Samples (d)'
ylabel 'Variance of Vega Estimator'
```

Filename: "digital_call.m"

This function [4] calculates the analytic [5] value/delta/vega for a digital call option.

```
% function V = digital_call(r,sigma,T,S,K,opt)
%
% Black-Scholes digital call option solution
%
% r     - interest rate
% sigma - volatility
% T     - time interval
```

```
% S     - asset value
% K     - strike price
% opt   - 'value', 'delta', 'gamma' or 'vega'
% V     - option value
%
function V = digital_call(r,sigma,T,S,K,opt)
if nargin ~= 6
  error('wrong number of arguments');
end
S  = max(1e-100,S);     % avoids problems with S=0
K  = max(1e-100,K);     % avoids problems with K=0
d2 = ( log(S) - log(K) + (r-0.5*sigma^2)*T ) / (sigma*sqrt(T));
switch opt
  case 'value'
    V = exp(-r*T).*N(d2);
  case 'delta'
    V = exp(-r*T)*(exp(-0.5*d2.^2)./sqrt(2*pi))./(sigma*sqrt(T)*S);
  case 'gamma'
    V = exp(-r*T)*(exp(-0.5*d2.^2)./sqrt(2*pi)) ...
             .*(-d2./(sigma*sqrt(T)*S) - 1./S) ./(sigma*sqrt(T)*S);
  case 'vega'
    V = exp(-r*T)*(exp(-0.5*d2.^2)./sqrt(2*pi)).*(-d2/sigma-sqrt(T));
  otherwise
    error('invalid value for opt -- must be ''value'',
          ''delta'', ''gamma'', ''vega''')
end

%
% Normal cumulative distribution function
%
function ncf = N(x)
%ncf = 0.5*(1+erf(x/sqrt(2)));
xr = real(x);
```

```
xi = imag(x);
if abs(xi)>1e-10
  error 'imag(x) too large in N(x)'
end
ncf = 0.5*(1+erf(xr/sqrt(2))) ...
    + i*xi.*exp(-0.5*xr.^2)/sqrt(2*pi);
```

Filename: "LRM.m"

This function calculates the delta and vega using the likelihood ration method, as well as the corresponding confidence interval bounds.

```
M     = 100000;
r     = 0.05;
sigma = 0.2;
T     = 1;
K     = 100;
S0    = 100;
W = sqrt(T)*randn(M,1);
S = S0*exp((r-0.5*sigma^2)*T+sigma*W);
f = exp(-r*T)*(0.5)*(1 + sign(S-K));
% delta
g = f .* W/(S0*sigma*T);
gbar = sum(g)/M;
sig2 = sum(g.^2)/M - gbar^2;
gexact = digital_call(r,sigma,T,S0,K,'delta');
disp(sprintf('Analytic Delta:  %f',gexact))
disp(sprintf('LRM Delta: %f  +/- %f',gbar,3*sqrt(sig2/M)))
% vega
g = f .* ( (W.^2-T)/(sigma*T) - W);
gbar = sum(g)/M;
sig2 = sum(g.^2)/M - gbar^2;
gexact = digital_call(r,sigma,T,S0,K,'vega');
disp(sprintf('Analytic Value:  %f',gexact))
```

```
disp(sprintf('LRM Vega: %f  +/- %f',gbar,3*sqrt(sig2/M)))
```

Filename: "vibratos.m"

This function displays (for either value, delta, or vega), the analytic value, the vibrato value calculated using low variance estimators, the vibrato value calculated using ordinary estimators, and the relevant confidence interval bounds.

```
% function [NLV, CILV, N, CI] = vibratos(r, sigma, T, S0, K, M, M2, N, opt)
%
% European digital call option with asset price following 1-dimensional
% geometric Brownian motion (Euler-Maruyama approximation). Calculation of
% Greeks using Vibrato Monte Carlo (both low variance estimators and
% ordinary estimators are considered). Confidence intervals are also given.
%
% r     - interest rate
% sigma - volatility
% T     - time interval
% S0    - initial asset value
% K     - strike price
% opt   - 'value', 'delta', 'vega'
% M     - number of Monte Carlo paths
% M2    - number of paths at a time
% N     - number of timesteps
% NLV   - numerical option value using low variance estimators
% N     - numerical option value using ordinary estimators
% CILV  - confidence interval for low variance estimators
% CI    - confidence interval for ordinary estimators

function [NLV, CILV, N, CI] = vibratos(r, sigma, T, S0, K, M, M2, N, opt)
if nargin ~= 9
    error('wrong number of arguments');
```

```
end
sum1 = 0;
sum1a = 0;
sum2 = 0;
sum2a = 0;
sum3 = 0;
sum3a = 0;
sumX = 0;
sumXa = 0;
sumY = 0;
sumYa = 0;
sumZ = 0;
sumZa = 0;
h = T/N;
for m = 1:M2:M
    m2 = min(M2, M-m);
    S = S0*ones(1,m2);
    dS = ones(1,m2);    % Used in delta calculation
    ds = zeros(1,m2);   % Used in vega calculation
    for n = 1:N-1
        dW = sqrt(h)*randn(1,m2);
        ds = ds.*(1 + r*h + sigma*dW) + dW.*S;   % For vega
        S = S.*(1 + r*h + sigma*dW);
        dS = dS.*(1 + r*h + sigma*dW);           % For delta
    end
    muw = S + r*S*h;
    sigw = sigma*S*sqrt(h);
    dmuw = (1+r*h)*dS;                          % For delta
    dsigw = sigma*sqrt(h)*dS;                   % For delta
    dmuws = (1+r*h)*ds;                         % For vega
    dsigws = sigma*sqrt(h)*ds + sqrt(h)*S;      % For vega
    Z = randn(1,m2);
    SN = muw + Z.*sigw;
```

```
    SNm = muw - Z.*sigw;
    fSN = (0.5)*(1 + sign(SN - K));
    fSNm = (0.5)*(1 + sign(SNm - K));
    fmuw = (0.5)*(1 + sign(muw - K));
    invsigw = sigw.^(-1);
    e1 = (0.5*Z.*invsigw).*(fSN-fSNm);
    e2 = (0.5*(Z.^2 - 1).*invsigw).*(fSN-2*fmuw+fSNm);
    e3 = fmuw + 0.5*(fSN-2*fmuw+fSNm);
    fdlog = e1.*dmuw + e2.*dsigw;
    sum1 = sum1 + sum(fdlog);
    sum1a = sum1a + sum((fdlog).^2);
    fdlogs = e1.*dmuws + e2.*dsigws;
    sum2 = sum2 + sum(fdlogs);
    sum2a = sum2a + sum((fdlogs).^2);
    sum3 = sum3 + sum(e3);
    sum3a = sum3a + sum((e3).^2);
    E1 = (Z.*invsigw).*fSN;
    E2 = ((Z.^2 - 1).*invsigw).*fSN;
    Fdlog = E1.*dmuw + E2.*dsigw;
    sumX = sumX + sum(Fdlog);
    sumXa = sumXa + sum((Fdlog).^2);
    Fdlogs = E1.*dmuws + E2.*dsigws;
    sumY = sumY + sum(Fdlogs);
    sumYa = sumYa + sum((Fdlogs).^2);
    sumZ = sumZ + sum(fSN);
    sumZa = sumZa + sum((fSN).^2);
end
% For delta (low variance estimators)
Delta = sum1/M;
delta = exp(-r*T)*Delta;
% For vega (low variance estimators)
Vega = sum2/M;
vega = exp(-r*T)*Vega;
```

```
% For value (low variance estimators)
Value = sum3/M;
value = exp(-r*T)*Value;
% Variances (low variance estimators)
sig_delta = sum1a/M - (sum1/M)^2;
sig_vega = sum2a/M - (sum2/M)^2;
sig_value = sum3a/M - (sum3/M)^2;
% Confidence intervals (low variance estimators)
CI_delta = 3*sqrt(sig_delta/M);
CI_vega = 3*sqrt(sig_vega/M);
CI_value = 3*sqrt(sig_value/M);
% For delta (ordinary estimators)
DDelta = sumX/M;
ddelta = exp(-r*T)*DDelta;
% For vega (ordinary estimators)
VVega = sumY/M;
vvega = exp(-r*T)*VVega;
% For value (ordinary estimators)
VValue = sumZ/M;
vvalue = exp(-r*T)*VValue;
% Variances (ordinary estimators)
Sig_delta = sumXa/M - (sumX/M)^2;
Sig_vega = sumYa/M - (sumY/M)^2;
Sig_value = sumZa/M - (sumZ/M)^2;
% Confidence intervals (ordinary estimators)
CCI_delta = 3*sqrt(Sig_delta/M);
CCI_vega = 3*sqrt(Sig_vega/M);
CCI_value = 3*sqrt(Sig_value/M);
switch opt
    case 'value'
        NLV = value;
        CILV = CI_value;
        N = vvalue;
```

```
        CI = CCI_value;
    case 'delta'
        NLV = delta;
        CILV = CI_delta;
        N = ddelta;
        CI = CCI_delta;
    case 'vega'
        NLV = vega;
        CILV = CI_vega;
        N = vvega;
        CI = CCI_vega;
    otherwise
        error('invalid value for opt; must be ''value'', ''delta'', ''vega''');
end
```

Filename: "estimator_comparison.m"

This code displays the output of the files digital_call.m and vibratos.m, (i.e. analytic vs. vibrato using low variance estimators vs. vibrato using ordinary estimators).

```
% Compares vibrato method using low variance estimators, to vibrato method
% using ordinary estimators.
r = 0.05;
sigma = 0.2;
T = 1;
S0 = 100;
K = 100;
M = 100000;
M2 = 1000;
N = 64;
h = T/N;
% Analytic value/delta/vega
```

```
value_exact = digital_call(r, sigma, T, S0, K, 'value');
delta_exact = digital_call(r, sigma, T, S0, K, 'delta');
vega_exact = digital_call(r, sigma, T, S0, K, 'vega');
% Vibrato Monte Carlo value/delta/vega
[value_NLV, value_CILV, value_N, value_CI] =
    vibratos(r, sigma, T, S0, K, M, M2, N, 'value');
[delta_NLV, delta_CILV, delta_N, delta_CI] =
    vibratos(r, sigma, T, S0, K, M, M2, N, 'delta');
[vega_NLV, vega_CILV, vega_N, vega_CI] =
    vibratos(r, sigma, T, S0, K, M, M2, N, 'vega');
% Comparison for Value
disp(sprintf('analytic value: %f', value_exact))
disp(sprintf('vibrato value using lower variance estimator:
    %f +/- %f', value_NLV, value_CILV))
disp(sprintf('vibrato value without lower variance estimator:
    %f +/- %f', value_N, value_CI))
% Comparison for Delta
disp(sprintf('analytic delta: %f', delta_exact))
disp(sprintf('vibrato delta using lower variance estimator:
    %f +/- %f', delta_NLV, delta_CILV))
disp(sprintf('vibrato delta without lower variance estimator:
    %f +/- %f', delta_N, delta_CI))
% Comparison for Vega
disp(sprintf('analytic vega: %f', vega_exact))
disp(sprintf('vibrato vega using lower variance estimator:
    %f +/- %f', vega_NLV, vega_CILV))
disp(sprintf('vibrato vega without lower variance estimator:
%f +/- %f', vega_N, vega_CI))
```

Filename: "timestep_size.m"

This file considers analytic vs. low variance vibrato Monte Carlo approximation vs. ordinary vibrato Monte Carlo approximation, for different values of

N, where N is the number of timesteps. As N increases, the timestep size, h, decreases. We are interested in the effect of this on the accuracy of the Monte Carlo estimates for the value, delta, and vega of the option.

```
% Looks at how a decrease in timestep size effects the numerical
% approximation given by the vibrato Monte Carlo method
%
r = 0.05;
sigma = 0.2;
T = 1;
S0 = 100;
K = 100;
M = 100000;
M2 = 1000;
% Analytic value/delta/vega
value_exact = digital_call(r, sigma, T, S0, K, 'value');
delta_exact = digital_call(r, sigma, T, S0, K, 'delta');
vega_exact = digital_call(r, sigma, T, S0, K, 'vega');
for N = 1:1:100
    h = T/N;
    % Numerical value/delta/vega and confidence interval, low variance
    % estimators and ordinary estimators
    [value_approx(N), CI_value_approx(N), value_approx2(N), CI_value_approx2(N)]
        = vibratos(r, sigma, T, S0, K, M, M2, N, 'value');
    [delta_approx(N), CI_delta_approx(N), delta_approx2(N), CI_delta_approx2(N)]
        = vibratos(r, sigma, T, S0, K, M, M2, N, 'delta');
    [vega_approx(N), CI_vega_approx(N), vega_approx2(N), CI_vega_approx2(N)]
        = vibratos(r, sigma, T, S0, K, M, M2, N, 'vega');
end
N = 1:1:100;
plot(N,value_exact, N, value_approx(N), N, value_approx2(N))
legend('Analytic Value', 'Vibrato Value (using low variance estimates)',
    'Vibrato Value (using ordinary estimates)')
```

```
title 'Comparison of Analytic Value and Vibrato Values'
xlabel 'Number of Timesteps (N)'
ylabel 'Option Value'
N = 1:1:100;
plot(N,delta_exact, N, delta_approx(N), N, delta_approx2(N))
legend('Analytic Delta', 'Vibrato Delta (using low variance estimates)',
    'Vibrato Delta (using ordinary estimates)')
title 'Comparison of Analytic Delta and Vibrato Deltas'
xlabel 'Number of Timesteps (N)'
ylabel 'Option Delta'
N = 1:1:100;
plot(N,vega_exact, N, vega_approx(N), N, vega_approx2(N))
legend('Analytic Vega', 'Vibrato Vega (using low variance estimates)',
    'Vibrato Vega (using ordinary estimates)')
title 'Comparison of Analytic Vegas and Vibrato Vegas'
xlabel 'Number of Timesteps (N)'
ylabel 'Option Vega'
N = 1:1:100;
plot(N, CI_delta_approx(N), N, CI_delta_approx2(N))
legend('Variance of Vibrato Delta (using low variance estimates)',
    'Variance of Vibrato Delta (using ordinary estimates)')
title 'Comparison of Variances for Vibrato Delta Calculations'
xlabel 'Number of Timesteps (N)'
ylabel 'Variance of Vibrato Option Delta'
N = 1:1:100;
plot(N, CI_vega_approx(N), N, CI_vega_approx2(N))
legend('Variance of Vibrato Vega (using low variance estimates)',
    'Variance of Vibrato Vega (using ordinary estimates)')
title 'Comparison of Variances for Vibrato Vega Calculations'
xlabel 'Number of Timesteps (N)'
ylabel 'Variance of Vibrato Option Vega'
```