Implied Volatility Estimation using Adjoint Monte Carlo Methods

Andrea Cangiani

St Hugh's College

University of Oxford

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science in Mathematical Modelling and Scientific Computing

September 2000

Contents

List of Figures

Chapter 1

Introduction

In their fundamental model for derivative pricing, Black and Scholes assume that the *volatility* of the underlying asset is a known constant or a known deterministic function.

This is one of the greatest and more studied limits of the model because it gives inconsistency between theoretical prices and market prices. In the model, the volatility σ is a property of the underlying whereas the *strike* E and *expiry* T are properties of the derivative only. Thus, all prices observed in the market (for different strikes and expiries) should be solutions of the Black-Scholes equation for a given *implied* volatility. This is invariably false, hence the inconsistency.

Many remedies have been studied such as stochastic volatility models, jump diffusion models and local volatility models. None of them gives a completely satisfactory and practical solution, so that the problem is still widely under discussion.

This thesis is based on the model of local (deterministic) implied volatility.

Dupire [5] proved that, if the full range of prices for vanilla call were available, there exists one and only one consistent volatility surface $\sigma(S,t)$ where S is the underlying price and t represents the time. This is called the *local volatility surface* since it is conditional to today's price of the underlying.

The inverse problem to find the local volatility surface from the market prices can be proved to be ill-posed. Moreover, the options are traded for a finite set of expiries and strike prices, so some interpolation technique has to be used to represent the volatility [9].

To match real prices with Black-Scholes prices an optimisation problem has to be solved as a function of some design variables which define the volatility surface. If methods such as the quasi-Newton BFGS [16] are used to solve the optimisation problem, it is important to have an efficient algorithm for gradient evaluation.

The aim of this thesis is to prove that a discrete adjoint approach (see [14]) can be used to calculate the gradient of the objective function which defines the optimisation problem, when the Black-Scholes equation is solved using Monte Carlo simulation.

The gradient with respect to one design variable can be found by solving the linearised equation corresponding to a unit perturbation in that particular design variable.

The solution of the linearised equation $\tilde{u}$ is found by solving the linear system

$$A\tilde{u} = f, \tag{1.1}$$

for some given matrix A and vector f. The linearised perturbation in the objective function is given by

$$\frac{\partial L}{\partial D}\tilde{u} =: g^T\tilde{u}, \tag{1.2}$$

where L is the objective function and D a particular design variable.

Let v be the solution of

$$A^T v = g. \tag{1.3}$$

Since

$$v^T f = v^T A\tilde{u} = (A^T v)^T\tilde{u} = g^T\tilde{u},$$

it is mathematically equivalent to solve (1.1) and evaluate (1.2) or solve (1.3) and evaluate $v^T f$. The former is the *direct* approach while the latter is called the *adjoint* approach.

When many design variables are involved, the adjoint approach is computationally more efficient, the reason being that following the direct approach we must solve (1.1) for many different values of f (one for each design variable) and then compute (1.2), while in the adjoint approach we need to solve (1.3) only once and then evaluate the vector dot product $v^T f$ for the different values of f.

The adjoint approach is presented in Chapter 3 on a model problem for the equation of diffusion in one dimension

$$\frac{\partial u}{\partial t} = \frac{\partial^2}{\partial x^2}(Du). \tag{1.4}$$

The concept of randomness is naturally related to the concept of diffusion. We say that there is diffusion when a group of particles spread out (according to a macroscopic law) while each particle moves randomly. This interpretation makes possible the application of the Monte Carlo method (which is itself based on a similar concept) to the solution of (1.4).

Having applied the adjoint Monte Carlo method to the equation of diffusion (1.4), the adaptation of the algorithm for implied volatility estimation is quite straightforward. This is presented in Chapter 3 on a simple problem in which the volatility depends only on the underlying through two design parameters.

The definition of local volatility surfaces is given in Chapter 1 together with the introduction to the application of Monte Carlo methods to derivatives pricing.

Chapter 2

Derivative pricing and the Monte Carlo Method

2.1 The Black-Scholes Model

We begin with a short introduction of the Black-Scholes model for derivative pricing.

A *derivative* is a financial instrument whose value depends on the value of other *underlying* financial instruments like, for example, traded securities [7]. We will focus on some simple derivatives like european call options on one asset, the main point being the explanation and application of the numerical methods.

A european call option is a contract between the *holder* and the *writer*. At *expiry* date T the holder has the right, but not the obligation, to purchase the underlying asset (exercise the option) at a price E which is prescribed in the contract; E is called *exercise* price or *strike* price.

Every mathematical model for derivative pricing has to include a model for the underlying. The Black-Scholes model assumes that the underlying follows a simple geometric Brownian motion, while the option pricing is based on a dynamic hedging strategy under the hypothesis of no-arbitrage.

The underlying asset is assumed to satisfy the stochastic differential equation

$$\frac{dS}{S} = (\mu - d)dt + \sigma dX, \tag{2.1}$$

where dX is a Wiener process, i.e. a random variable drawn from a normal distribution with mean 0 and variance dt.

The parameter μ measures the expected growth rate which depends on the risk-free interest rate r and also on the risk of the return from the asset.

If the asset pays a dividend, which is in general assumed to be known in advance, this also enters the deterministic part of equation (2.1) trough the parameter d.

The value of a derivative is independent of μ. On the contrary, the volatility σ, which measures the standard deviation of the rate of change of S, is of a crucial importance for derivative pricing.

If S follows (2.1), then it has lognormal distribution, i.e. its natural logarithm is normally distributed. This can be proved using Itô's Lemma which is an extension of Taylor's Formula to functions of stochastic variables.

Lemma 1 (Itô) *If f is a function of the stochastic variable S and S follows the Itô process*

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

then

$$df = f'(S)(a\,dt + b\,dX) + \frac{1}{2}b^2 f''(S)dt.$$

If S satisfies (2.1), applying Itô's Lemma to $f(S) = \log S$ we get

$$d(\log S) = (\mu - d - \frac{1}{2}\sigma^2)dt + \sigma dX, \tag{2.2}$$

so $\log S$ has normal distribution (with drift $(\mu - d - \frac{1}{2}\sigma^2)t$ and variance $\sigma^2 t$) since it follows a constant coefficient random walk (and dX is normal).

We can easily solve (2.2) to find S:

$$\log(S(t)) = \log(S(t_0)) + (\mu - d - \frac{1}{2}\sigma^2)t + \sigma X(t) \Rightarrow$$

$$S(t) = S(t_0)e^{(\mu - d - \frac{1}{2}\sigma^2)t + \sigma X(t)} \tag{2.3}$$

Thus, we have an exact expression for the value of the asset at any time in the future in function of the random variable X.

As we will see later, in pricing with Monte Carlo simulation, it is convenient to use the exact expression (2.3). In particular, the value V of non path-dependent options can be approximated simulating the underlying price using (2.3) through only one time step $T - t_0$.

With this as model for the underlying, we come to the pricing of the option.

The derivation of the Black-Scholes equation assumes that the value of the option depends only on the current price of the underlying S and time t, so that we write $V = V(S,t)$.

Under the additional assumptions:

- no-arbitrage (no opportunities to make instantaneous risk-free profit),

- existence of risk-free investments (at known interest rate r).
- possibility of continuous hedging,

the following equation is derived for the value of the option from a continuous binomial model, hedging the portfolio $\Pi = V - \Delta S$ to eliminate the risk

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-d)S\frac{\partial V}{\partial S} - rV = 0. \tag{2.4}$$

This is the basic Black-Scholes equation for option pricing. In (2.4) the parameters r and d are assumed to be constant or at least known in advance as functions of t. The volatility σ is assumed to be constant: this is the main limitation of the model since in practice the volatility is observed to be function of both S and t (or, more precisely, of E and T). This fact makes the model inconsistent with real prices. We will discuss this crucial problem at the and of the Chapter.

The vanilla call is modelled by (2.4) through the final condition (payoff)

$$V(S,T) = \max(S(T) - E, 0)$$

and boundary values

$$V(0,t) = 0, \quad V(S,t) \sim S \quad \text{as } S \to \infty.$$

The boundary and final value problem so obtained has been solved analytically and the solution, in the case of no dividends, is given by

$$V(S,t) = SN(d_1) - Ee^{-r(T-t)}N(d_2), \tag{2.5}$$

where

$$\begin{aligned} d_1 &= \frac{\log(S/E) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \\ d_1 &= \frac{\log(S/E) + (r - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \end{aligned}$$

and $N(\cdot)$ is the cumulative distribution function for a standardised normal random variable, which is given by

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}y^2} dy.$$

The drift μ is not present in the Black-Scholes equation (2.4): this is a consequence of the elimination of risk, which is present only through S, obtained with continuous hedging. This explains why we are able to solve analytically the equation to find today's value of the option as a function of the current price of the underlying.

In other words, in the Black-Scholes model, the option is priced as if the underlying followed the risk-neutral random walk

$$\frac{dS}{S} = rdt + \sigma dX. \tag{2.6}$$

2.2 Solving the Black-Scholes problem with Monte Carlo simulation

The distribution of maturity values of the option can be obtained from the distribution of the terminal stock value. This fact, together with the assumption of risk-neutrality, is crucial to the numerical price valuation via Monte Carlo simulation.

The expected return of the stock is the risk-free rate r, so the present value of the option can be obtained discounting the expected payoff at the risk-free rate of interest. Thus, the value at t is given by

$$V(S,t) = e^{-\int_t^T r(\tau)d\tau} E[V_T]. \tag{2.7}$$

Here, E denotes the expected value in a regime of risk-neutrality.

The Monte Carlo method is used to obtain the expected value $E[V_T]$.

A large number J of simulations of Brownian motion for the price of S are carried out starting from the present value. The payoffs for the option are calculated using the final values $S_j(T)$.

We will now see why this gives, as $J \to \infty$, the correct expected value for the option.

Let $\{V_j\}_{j=1}^{J}$ be the values of the option obtained discounting the payoffs resulting from the simulations of S.

Clearly, $E[V_j] = V$ for any j and the V_j's have all the same variance

$$(\bar{\sigma})^2 = E[(V_j - V)^2].$$

The Monte Carlo approximation to the expected value of the option is

$$\hat{V} = \frac{1}{J}\sum_{j=1}^{J} V_j$$

and, since V_j are independent random variables $\hat{V} \sim N(V, \hat{\sigma}^2)$ for some $\hat{\sigma}$ to be determined.

Let $\xi_j = \bar{\sigma}^{-1}(V_j - V)$, so that, since the V_j's are independent,

$$E[\xi_j] = 0, \quad E[\xi_j^2] = 1, \quad E[\xi_i \xi_j] = 0 \quad \text{if } i \neq j.$$

The Monte Carlo error can be written

$$\varepsilon_J = \hat{V} - V = \frac{1}{J}\sum_{i=1}^{J} V_j - V = \frac{\bar{\sigma}}{J}\sum_{j=1}^{J} \xi_j.$$

Using the properties of the ξ_j's just mentioned, we can work out the root mean square error or *standard error*

$$\begin{aligned}
\hat{\sigma} &= \operatorname{var}(\varepsilon_J)^{1/2} = \Big(E\Big[\Big(\frac{\bar{\sigma}}{J}\sum_j \xi_j\Big)^2\Big] + E\Big[\frac{\bar{\sigma}}{J}\sum_j \xi_j\Big]^2\Big)^{1/2} \\
&= \frac{\bar{\sigma}}{J}\Big\{E\Big[\sum_j \xi_j^2\Big] + E\Big[\sum_j \sum_{i \neq j} \xi_i \xi_j\Big]\Big\}^{1/2} \\
&= \frac{\bar{\sigma}}{J}\Big\{JE\big[\xi_j^2\big]\Big\}^{1/2} \\
&= \frac{\bar{\sigma}}{J^{1/2}}. \qquad (2.8)
\end{aligned}$$

This is a typical result for Monte Carlo methods: the standard error size is $O(J^{-1/2})$ with a constant which is the square root of the variance.

Since the variance is unknown (it is as difficult to compute as the value itself), to establish the accuracy of the solution the following empirical estimate is used for the variance:

$$\hat{\sigma}^2 = \frac{1}{J-1}\sum_{j=1}^{N}(V_j - \hat{V})^2.$$

For example, a 95% confidence interval for the option price is given by

$$\hat{V} - 1.96\,\hat{\sigma} < V < \hat{V} + 1.96\,\hat{\sigma}.$$

Moreover, it can also be proved (see [6]) that the error as a function of J has Gaussian distribution, i.e.

$$\hat{\sigma} \approx \frac{\bar{\sigma}}{J^{1/2}}\,\nu, \qquad (2.9)$$

where ν is standard normal ($N(0,1)$). More precisely,

$$\lim_{J\to\infty} \operatorname{Prob}\Big(a < \frac{J^{1/2}}{\bar{\sigma}}\hat{\sigma} < b\Big) = \operatorname{Prob}(a < \nu < b) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\,dx.$$

Thus, the error is of $O(J^{-1/2})$, but the statistical distribution of the error is approximately a normal random variable. In fact, (2.9) does not provide the absolute upper bound on the error typical of numerical methods such as finite differences and finite elements; instead it gives that the error is of a certain size with some probability [10]. Actually, it is a characteristic of stochastic methods like Monte Carlo to give probabilistic error bounds.

Compared with the typical rate of convergence of methods like the ones mentioned above or like the binomial method, the rate of convergence of Monte Carlo simulation is quite slow and decelerates in relation to the computational effort.

On the other hand, a Monte Carlo simulation is very easy to implement, it is more flexible then methods requiring grids and is robust in that its accuracy depends only on the crudest measure of the complexity of the problem. Indeed the attractive feature of Monte Carlo simulation is that the rate of convergence is independent of the dimension of the problem. Since the need of valuation of complicated portfolios of options leads to high dimensional mathematical problems, that characteristic has made Monte Carlo simulation one of the most popular numerical methods in this field.

Moreover, while when working in a high dimension with grid-based methods it becomes practically impossible to improve the accuracy because of the difficulty of refining the mesh, in principle every new Monte Carlo simulation improves the accuracy. The reason for this is that every simulation is itself an estimate of the correct price.

Having said this, it remains of crucial importance to find ways to accelerate the convergence rate of Monte Carlo simulation. Indeed, this is the main point discussed in the literature on Monte Carlo methods.

There are two ways to accelerate a Monte Carlo method. We can either reduce the constant of convergence or improve the rate of convergence itself.

Techniques of the first kind are known as *variance reduction methods* and the following two are examples of such methods.

2.2.1 Control variates

The idea which characterises this method is: use the solution of a simpler problem to improve the accuracy.

To valuate the price of an option V, we consider a second option V_1 (*control variate*), similar to V in nature, but for which the analytical price is available.

The Monte Carlo estimate will be given by

$$\hat{V}^{cv} = \hat{V} + (V_1 - \hat{V}_1),$$

where the known error $V_1 - \hat{V}_1$ is used as a *control* in the estimation of V.

The gain in efficiency will be measured by the reduction in the variance of $\hat{V}^{cv}$ as compared to the variance of $\hat{V}$. But this method involves the evaluation of two estimates ($\hat{V}^{cv}$ and $\hat{V}$), so we have to take in consideration the extra computational work.

In fact, the best results are obtained by pricing the two options using identical simulations for the underlying and in this case the additional work is moderate.

The relation between the variances is given by

$$\text{var}(\hat{V}^{cv}) = \text{var}(\hat{V}) + \text{var}(\hat{V}_1) - 2\,\text{cov}(\hat{V}, \hat{V}_1),$$

so the control variate estimate will have reduced variance provided that

$$\text{var}(\hat{V}_1) < 2\text{cov}(\hat{V}, \hat{V}_1).$$

To optimalize the method we can define the control variate estimate as

$$\hat{V}^{\beta} = \hat{V} + \beta(V_1 - \hat{V}_1),$$

where β is a parameter. The variance is now given by

$$\text{var}(\hat{V}^{\beta}) = \text{var}(\hat{V}) + \beta^2\text{var}(\hat{V}_1) - 2\beta\,\text{cov}(\hat{V}, \hat{V}_1)$$

and the variance can be minimised setting

$$\beta^* = \frac{\text{cov}(\hat{V}, \hat{V}_1)}{\text{var}(\hat{V}_1)}.$$

Using β^* we are guaranteed not to increase the variance. The problem is that β^* is not readily available since $\text{cov}(\hat{V}, \hat{V}_1)$ is unknown, but it can be estimated using the regression technique from the J estimates of the option values obtained from the simulation runs [17].

In general, in order to obtain variance reduction, the control variate has to be close enough to the option being estimated.

This requirement is often in antagonism with the necessity for the control variate to be analytically evaluable.

As far as option price evaluation is concerned, there are many options for which the analytical solution is available, so it is relatively simple to define good control variates. Moreover, the control variate can be really effective since it allows to get rid of the slow convergence due to the characteristic kink present in most option's payoffs (see figure 2).

2.2.2 Antithetic variables

The Monte Carlo simulation of paths for the underlying price is carried out generating finite sequences of standard normal random variables $\{\boldsymbol{\nu}_j\}_{j=1}^J$.

If $\boldsymbol{\nu}_j$ is standard normal, so is the *antithetic variable* $-\boldsymbol{\nu}_j$. Thus, $-\boldsymbol{\nu}_j$ can be used to generate a new random walk and a new option estimate $\tilde{V}_j$.

The values V_j and $\tilde{V}_j$ are expected to be negatively correlated in that, if one overshoots the correct value, the other undershoots it. Thus, it seems sensible to consider their average as an estimate for the option value, which will be given by

$$\hat{V}^{av} = \frac{1}{J}\sum_{j=1}^{J}\frac{V_j + \tilde{V}_j}{2}.$$

Since V_j and $\tilde{V}_j$ have the same variance,

$$\text{var}\Big[\frac{V_j + \tilde{V}_j}{2}\Big] = \frac{1}{2}(\text{var}[V_j] + \text{cov}[V_j, \tilde{V}_j]).$$

Thus, we have a variance reduction ($\text{var}[V^{av}] \leq \text{var}[\hat{V}]$) if

$$\text{cov}[V_j, \tilde{V}_j] \leq \text{var}[V_j].$$

Actually, since the computational work is approximately doubled, to assure an improvement in efficiency we require $\text{var}[\hat{V}^{av}] \leq \text{var}[\hat{V}]$ and so

$$\text{cov}[V_j, \tilde{V}_j] \leq 0. \tag{2.10}$$

It is not difficult to prove (see [12]) that when pricing a large range of options (including Asian options) condition (2.10) is satisfied.

In general, we should expect an improvement in accuracy from the use antithetic variables because the random samples related to the antithetic pairs are more regularly distributed then a collection of $2J$ independent samples. For example, the mean over the antithetic pairs is always 0, whereas the mean of finitely many independent samples is almost surely different from 0. Of course one should always take care when introducing correlations between the samples. For instance, the standard error has to be estimated using the standard deviation of the J averaged pairs and not the $2J$ individual samples.

2.2.3 Quasi-Monte Carlo methods

The second approach to acceleration of Monte Carlo is to modify the properties of the random sequences to improve the typical convergence rate of the (crude) Monte Carlo.

The basis for this approach is the observation that any sequence of random numbers is not quite uniformly distributed. The nonuniformity spoils the accuracy so it seems reasonable that convergence may be improved using more uniformly distributed numbers.

The deviation of a set of points from uniformity is measured by the notion of *discrepancy*.

Quasi-Monte Carlo methods use low-discrepancy (highly uniform) sequences which are a deterministic alternative to random numbers. Many quasi-random number generators have been developed with the nice property that when successive points are added the entire sequence remains at a similar level of discrepancy.

Because of the correlations induced to obtain uniformity, these methods are less versatile and so they are more suitable for integration (the typical convergence rate is $O(\frac{(\log J)^d}{J})$, where d is the dimension of the integral) than for simulation [10].

In our implementations, we used variance reduction techniques rather than quasi-Monte Carlo methods.

2.3 Implementation

As described in Section 1.2, under the assumption of risk-neutrality, the value V of an option is obtained discounting the expected value of the payoff.

The Monte Carlo method can be used to estimate the expected value of the random variable of Eq. (2.6).

For the vanilla call (2.5), the expected payoff at expiry is

$$E[\max(S(T) - E, 0)] \tag{2.11}$$

and, the value is found discounting it to the present time t:

$$V = e^{-r(T-t)} E[\max(S(T) - E, 0)].$$

(For simplicity here and in the sequel we are to assume constant risk-free rate). To find numerically the expectation (2.11), the underlying price is simulated using a discrete-time version of the risk neutral model (2.6) such as the Euler method

$$\Delta S = rS\Delta t + \sigma\sqrt{\Delta t}\,\nu, \tag{2.12}$$

where ν is drawn from a standardised normal distribution. The simulation is carried out starting from today's price and generating $N = (T-t)/\Delta t$ random numbers:

$$\begin{aligned} S^0 &= S(t) \\ S^{n+1} &= S^n + rS^n\Delta t + \sigma S^n\sqrt{\Delta t}\,\nu^n, \quad n = 0,\dots N-1. \end{aligned}$$

This method is first order in time.

Alternatively, as long as σ is constant so that S is lognormally distributed, we can refer to (2.3) and use

$$S_{t+\Delta t} = S_t\, e^{(r-\frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\,\nu}. \tag{2.13}$$

In this case no error in t is introduced unless the option priced is path-dependent.

Figure 2.1 shows some random walks generated using (2.13) with $t = 0$, $T = 1$ and $N = 500$.

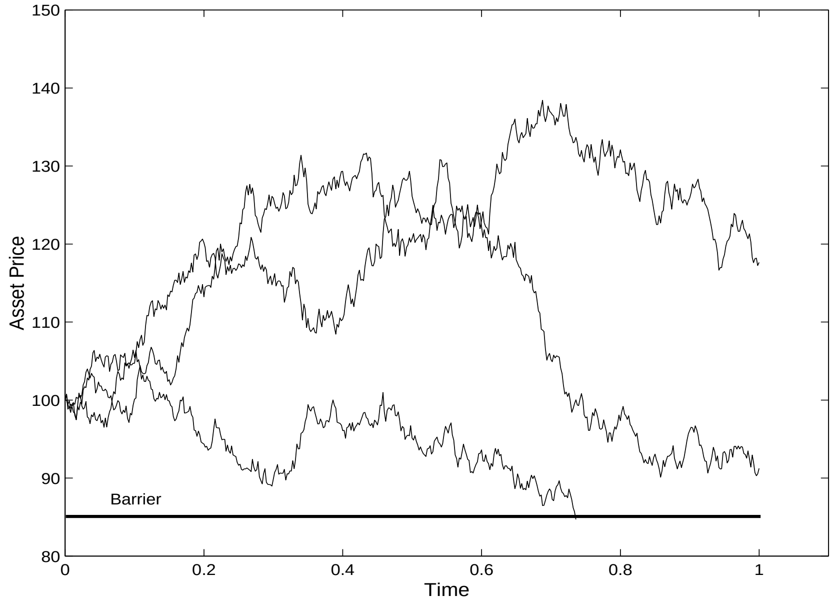

Figure 2.1: Sample paths for the underlying generated as 500 steps random walks.

To carry out the random walks we need to generate sequences of random numbers. Actually, the computational generation of good samples of random numbers (pseudo-random numbers) is the main problem of Monte Carlo numerical methods since the implementation is straight-forward and no stability analysis is needed. There are many good generators of uniformly distributed real random numbers, like linear congruental methods [3]. We have used the MATLAB random number generator, just because it is well tested and fast. But we are aware that, in particular when working in high dimension, ad hoc random number generators have to be used in order to avoid correlations.

Random numbers distributed according to a desired density $p(x)$ can be obtained from uniformly distributed random numbers. This can be done via a general procedure which involves the inversion of the cumulative distribution function associated with p [2]. In same cases it is possible to derive special transformations cheaper then the general transformation method. A very famous and widely used method for generating standardised normal samples is the Box-Muller Transform. Given two uniform independent samples u and v between 0 and 1,

$$\begin{aligned} x &= \sqrt{-2\log u}\cos(2\pi v), \\ y &= \sqrt{-2\log u}\sin(2\pi v) \end{aligned}$$

are independent Gaussian samples.

We have tested our MATLAB code evaluating simple option prices like vanilla call or path-dependent options like down-and out call for which the analytical price is readily available. Figure 2.2 shows the typical payoff diagram for a vanilla call and some values of the option evaluated for different spot prices for the underlying using Monte Carlo simulation.

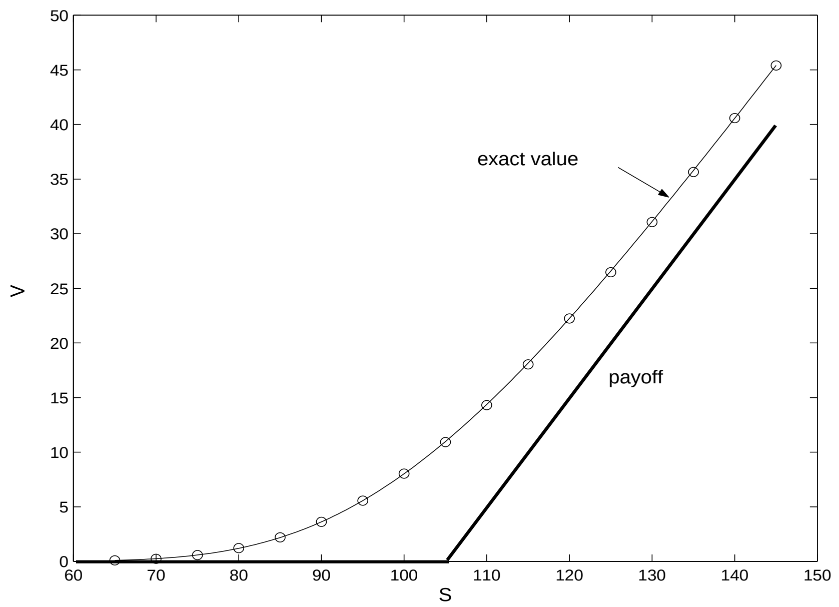

Figure 2.2: Monte Carlo and exact (a-dimensional) values of a vanilla call versus the spot price S. The strike price is $E = 105$ and $T = 1$.

The exact solution for the vanilla call is given by (2.5) and can be easily calculated in MATLAB which has its own function *erfc.m* for the complementary error function.

The convergence diagram is presented in Figure 2.3.

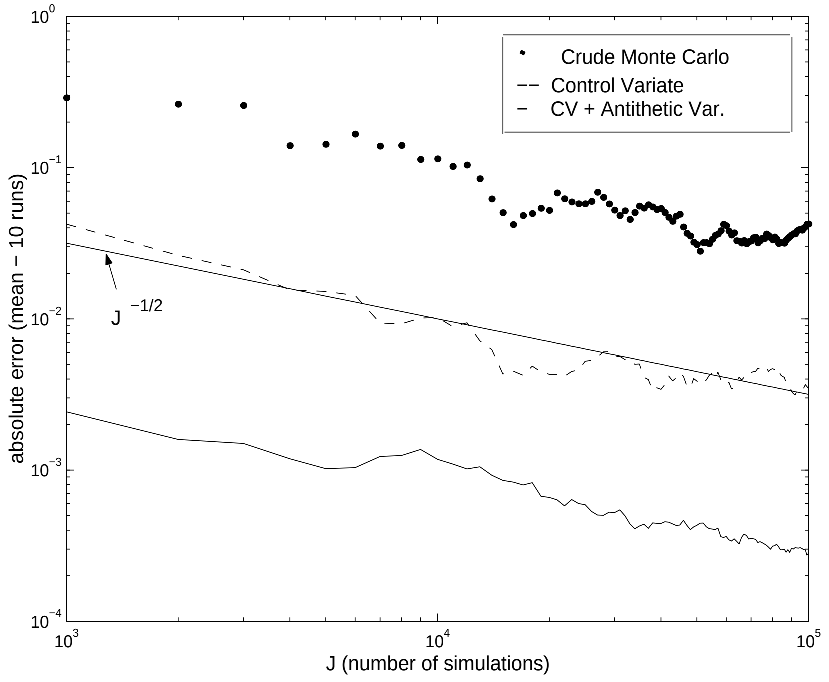

Figure 2.3: Loglog plot of the error in the evaluation of a vanilla call (S=100, E=105, T=1). The rate of convergence is the theoretical $N^{-1/2}$, but the constant of convergence is reduced using variance reduction techniques.

This shows the absolute error $|\hat{V}_J - V|$ in function of the number of simulations J for crude Monte Carlo, control variate (where a call with different strike was used as control variate) and antithetic variable. Notice that, since the convergence rate is of statistical nature, to show the qualitative behaviour of the error we have plotted its average over a number of runs.

The convergence rate is confirmed to be of $O(J^{-1/2})$ for all methods, but a significant error reduction is achieved in particular when both of the variance reduction techniques are used. This shows that the error reduction achieved by the two methods is of different nature. Further improvement will be obtainable combining variance reduction and quasi-Monte Carlo methods.

Monte Carlo methods can be used to price most kinds of options, although, because of the backward nature of the early exercise, American option evaluation is not straight-forward (see [17] for a list of algorithms).

A simple example of path-dependent option on which we tested our code is the down-and-out vanilla call. This has the same payoff of a vanilla call but with a different boundary condition: the option expires worthless if a barrier B is reached by the asset from above at any time before expiry. (These kinds of options are said to be 'weakly path-dependent' since they can still be evaluated from the current values S and t). To check our results we compare them with the analytical solution which can be expressed in terms of vanilla calls:

$$V_{d/o}(S,t) = V(S,t) - (S/B)^{2\alpha} V(B^2/S,t),$$

where

$$\alpha = \frac{1}{2}(1 - 2\frac{r}{\sigma^2})$$

The Mote Carlo and analytic price of a down-and-out call option with barrier $B = 85$ strike $E = 105$ is plotted in Figure 2.4 in function of today's price of the underlying.

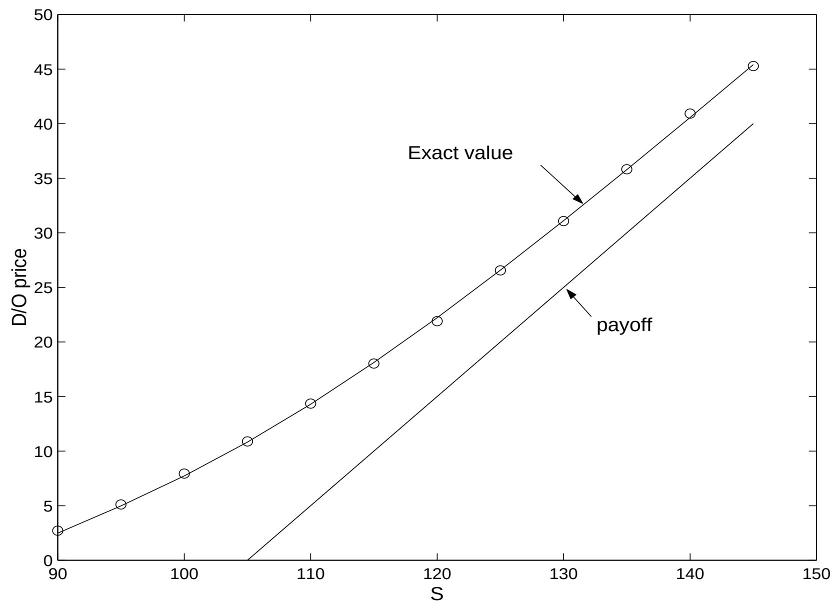

Figure 2.4: Monte Carlo and exact values of a down-and-out call versus the spot price S.

2.4 Local implied volatility surfaces

So far, according to the classical Black-Scholes analysis, we where dealing with constant volatility.

In the Black-Scholes model, σ is a property of the underlying whereas the strike E and expiry T are properties of the option only. Thus, if the model were consistent, all prices observed in the market for different strikes and expiries should be solutions of the Black-Scholes equation for a given volatility.

Counterposing this reasoning, i.e. inverting the equation, the real prices should give one and only one Black-Scholes *implied volatility.*

This however is not the case in practice.

A first problem is that the inverse problem for a given option price has a unique solution (the implied volatility) only if the *vega*

$$\frac{\partial V}{\partial \sigma}(S,t)$$

does not change sign for any value of S, t. This is true, for example, for vanilla calls and puts, but is not true in general. For instance, the vega of a cash-or-nothing call does change sign, thus there can be more possible values for the implied volatility.

A second problem is that the implied Black-Scholes volatility strongly depends on both the expiry and the strike price of the market prices.

The dependence on expiry can be treated defining a (*term-structure*) for the volatility. The Black-Scholes equation becomes

$$V_t + \frac{1}{2}\sigma^2(t)S^2V_{SS} + (r(t) - d(t))SV_S - r(t)V = 0. \tag{2.14}$$

This can be easily transformed into a constant coefficient equation ([HOWISON]), so that the solution of (2.14) can be given solving the constant parameter equation (2.4) using

$$\begin{aligned} r_c &= \frac{1}{T-t}\int_t^T r(\tau)d\tau, \\ \sigma_c &= \frac{1}{T-t}\int_t^T \sigma(\tau)d\tau, \\ d_c &= \frac{1}{T-t}\int_t^T d(\tau)d\tau. \end{aligned}$$

Thus, given real prices of vanilla calls for all expiry dates T, we should be able to solve an inverse problem to find the deterministic time-dependent volatility which will be used to price more complicated contracts such as American and path-dependent options.

In reality, the options are traded in the market for a finite set of expiries (usually three, six and nine months), so we do not have the whole range of prices from which to deduce the implied volatility and some kind of interpolation technique is needed. This is done in practice by many trading companies, for example assuming that the volatility is a piecewise linear function of t.

The dependence of the implied volatility on the strike price is more difficult to determine in that it is definitely inconsistent with the Black-Scholes model and is often referred to as the *smile* effect.

One way of extending the model is to assume that the volatility is a deterministic function of both the spot price and the time:

$$\sigma = \sigma(S, t).$$

The question is whether this is possible. And, if yes, is σ unique and valid at anytime in the future? We can reformulate this questions as follows.

In absence of dividends and assuming a constant risk-free rate, an option should still verify the Black-Scholes equation

$$V_t + \frac{1}{2}\sigma^2(S,t)S^2 V_{SS} + rSV_S - rV = 0. \tag{2.15}$$

The solution can again be written as the discounted expected value of the payoff, so

$$V(S,t) = e^{-r(T-t)} \int_0^\infty p(S,t;S',T)V(S',T)dS',$$

where $p(S,t;S',T)$ is the risk-neutral probability density associated with the Kolmogorov or Fokker-Planck equations

$$\frac{\partial p}{\partial T} = \frac{1}{2}\frac{\partial^2}{\partial S'^2}(\sigma(S',T)^2 S'^2 p) - \frac{\partial}{\partial S'}(rS'p), \tag{2.16}$$

$$-\frac{\partial p}{\partial t} = \frac{1}{2}\sigma(S,t)^2 S^2 \frac{\partial^2 p}{\partial S^2} + rS\frac{\partial p}{\partial S}. \tag{2.17}$$

In the forward equation (2.16), p is regarded as the probability density that at T the price will be S' given that the present value is S (conditional probability). On the other hand, in the backward equation (2.17), p is the (conditional) probability density that at t the price was S given that the price at T is S'.

For a call option with expiry T and strike E, the value is given by

$$V(S,t;E,T) = e^{-r(T-t)} \int_E^\infty p(S,t;S',T)(S'-E)dS'.$$

The particular form of the payoff allows us to differentiate twice in E and to get p as a function of V:

$$\frac{\partial V}{\partial E} = -e^{-r(T-t)} \int_E^\infty p(S,t;S',T)dS'$$

and

$$\frac{\partial^2 V}{\partial E^2} = e^{-r(T-t)} p(S,t;E,T),$$

so

$$p(S,t;E,T) = e^{r(T-t)} p(S,t;E,T). \qquad (2.18)$$

Thus, the market price of the vanilla call (with given E and T) is consistent with Black-Scholes if in the model we use the probability density given by (2.18) as distribution of the price of the asset.

Thus the question becomes: is there a unique *implied* diffusion process (and so a unique $\sigma(S,t)$) consistent with all market prices? In other words, can we think of $p(S,t;E,T)$ with S, t fixed and E, T variable as the density function (conditional to the spot price) consistent with today's market view of vanilla calls?

Using the duality of the forward and backward Kolmogorov equations, Dupire [5] proved that there does exist a unique $\sigma(E,T)$ consistent with $V(S,t;E,T)$.

Relabelling S, t with S_1, t_1 and E, T with S, t this gives the volatility

$$\sigma(S,t) = \sigma(S_1, t_1; S, t) \qquad (2.19)$$

contingent to today's datas S_1, t_1.

This is known as the *local volatility surface.* (A different approach due to Shimko [15] is to use the probability density (2.18) to predict option prices.)

It is worth remarking that (2.19) is conditional on today's spot price, it is consistent with today's prices but it cannot be used to infer tomorrow's prices. Moreover, the local volatility surface is determined from given vanilla prices, thus in adopting this method we renounce to price these options.

However, the implied volatility can be used to price exotic options consistently [5] and then, as the time goes on, it can be used to hedge positions on such contracts using vanilla options and the new volatility surface.

In practice, since only few prices are known, a numerical method has to be used to fit parametric volatility surfaces to the datas.

Many methods has been proposed (see [9], [5] as references) to solve this inverse problem.

We propose to adopt a discrete adjoint method to work out the gradient sensitivities needed by minimisation methods using gradients.

Chapter 3

Diffusion problems

3.1 Continuous random walk

A mathematical law describing some physical or biological phenomenon can often be obtained either by following a deterministic (macroscopic) approach, or as the result of averaging some stochastic (microscopic) behaviour.

The Black-Scholes equation, a type of diffusion equation, is derived via an hedging argument based on the continuous-time limit of the binomial model (Brownian motion) for the underlying. The behaviour of the asset is stochastic, but its final distribution is prescribed and gives the option price as an expectation. Thus, we are naturally led to a deterministic equation from the stochastic model.

Of course, the constant coefficient Black-Scholes equation is strictly related to the diffusion equation which describes simple diffusion phenomena (the two equations are the same up to a change of variable).

The diffusion equation is readily obtained via the classical approach to diffusion: the flux is proportional to the gradient of the concentration (Fickian diffusion). In one dimension

$$\frac{\partial U}{\partial t} = D\frac{\partial^2 U}{\partial x^2},$$

where D is the coefficient of diffusion or diffusivity. The same equation can be derived from a law of particles motion.

In general, a diffusion phenomenon arises when the irregular motion of each of a large number of particles or individuals gives rise to a regularity of motion of the whole group. Moreover, the long-term statistical trend of the random motion of a single particle is observed to follow the same deterministic law.

We can derive the diffusion equation from a binomial model for particle motion.

Suppose that every particle follows a random walk moving with equal probability to the left or to the right a fixed distance Δx over the time Δt (we say that the random walk is unbiased or isotropic: using biased random walk we will get an extra advective term).

The probability that a particle will move m steps either to the left or to the right after n steps is known to be given by the binomial (or Bernoulli) distribution

$$p(m,n) = \frac{1}{2^n} \frac{n!}{((n+m)/2)!((n-m)/2)!}.$$

This distribution converges, as n approaches infinity, to the Gaussian (or normal) distribution as

$$p(m,n) \sim \left(\frac{2}{\pi n}\right)^{1/2} e^{-m^2/2n}, \quad \text{as } n \to \infty.$$

Moreover, if $x = m\Delta x$ and $t = n\Delta t$ are considered to be continuous variables and we let $\Delta x, \Delta t \to 0$ prescribing that

$$\lim_{\substack{\Delta x \to 0 \\ \Delta t \to 0}} \frac{(\Delta x)^2}{2\Delta t} = D, \tag{3.1}$$

the probability density function of the associated continuous random walk is

$$p(x,t) = \frac{1}{(4\pi D t)^{1/2}} e^{-\frac{x^2}{4Dt}}. \tag{3.2}$$

(Of course this is also a solution of the diffusion equation with constant diffusivity D!)

Thus, in the continuous model the particles spread out following a Gaussian distribution.

The choice given by (3.1) could appear arbitrary. Actually, that choice is the only one that gives nontrivial probability density and is the defining characteristic of Brownian motion. Notice that, in the limit, the discrete speed of walk $c = \Delta x/\Delta t$ approaches infinity.

To derive the diffusion equation from the binomial model, we consider the probability of a particle to be in x at t:

$$p(x,t) = \frac{1}{2} p(x - \Delta x, t - \Delta t) + p(x + \Delta x, t - \Delta t).$$

Expanding the right hand side in Taylor series we get

$$\frac{\partial p}{\partial t} = \left(\frac{\Delta x^2}{2\Delta t}\right) \frac{\partial^2 p}{\partial x^2} + \left(\frac{\Delta t}{2}\right) \frac{\partial^2 p}{\partial^2 t} + \Delta x \frac{\partial^2 p}{\partial x \partial t} + \cdots .$$

If we now let $\Delta x, \Delta t \to 0$ according to (3.1), we obtain

$$\frac{\partial p}{\partial t} = D\frac{\partial^2 p}{\partial x^2}. \tag{3.3}$$

Finally, let J be the total number of particles so that $U(x,t) = Jp(x,t)$ is the concentration of particles. From (3.3) we have that the concentration satisfies the one-dimensional diffusion equation with constant diffusivity

$$\frac{\partial U}{\partial t} = D\frac{\partial^2 U}{\partial x^2}\,. \tag{3.4}$$

We will consider the generalisation of (3.4)

$$\frac{\partial U}{\partial t} = \frac{\partial^2}{\partial x^2}\Big(D(x,t)U\Big) \tag{3.5}$$

which is of the same form as the forward Kolmogorov equation (2.16). This can still be derived from a random walk model in which a step depends on the state at the point of departure but not on the conditions at the point of arrival.

Notice that the heat phenomenon is of a different nature and indeed it is described by a different flux and equation. The heat is always directed in the direction of lower temperature. Thus, the motion of a single particle does depend on the surrounding characteristics and not only on its initial position. See [11] for a full account on diffusion phenomena and their random walk interpretation.

Having related the diffusion equation to the random walk approach we are ready to apply Monte Carlo simulation to solve the equation numerically.

3.2 Model problems and Monte Carlo simulation

The continuous random walk model just discussed states that a particle follows Brownian motion. The particle position $X = X(t)$ is regarded as a time-dependent random variable.

Formally, a random variable $X(t)$ follows Brownian motion if

1. for each $t > 0$ and $h > 0$, $X(t+h) - X(t)$ is normal with mean 0 and variance h,
2. $X(t+h) - X(t)$ is independent of $X(t)$,
3. $X(0) = 0$.

The Monte Carlo approximation is based on the simulation of a finite number of particle trajectories which represent discretizations of the Brownian motion. We will now see how this can be done in practice with the help of some examples.

3.2.1 Infinite domain

Consider the simple problem

$$\begin{cases} \dfrac{\partial U}{\partial t} = D\dfrac{\partial^2 U}{\partial x^2}, & x \in \mathcal{R},\ 0 < t \leq T, \\ U(x,0) = \delta(x). \end{cases} \tag{3.6}$$

for which the solution is well-known to be of the form (3.2).

We consider J particles starting from the initial position $X_j^0 = 0$, $j = 1, \ldots, J$. Over any non infinitesimal time-interval δt, the Brownian motion is a normal random variable. Thus, the random walks are given by

$$X_j^{n+1} = X_j^n + \delta X, \quad n = 0, \ldots, N-1,$$

where $\delta X \sim N(0, 2D\delta t)$.

In practice, we generate random numbers normally distributed with mean 0 and variance 1, thus the simulation algorithm gives the particles final positions through

$$\begin{cases} X_j^0 = 0, \\ X_j^{n+1} = X_j^n + \sqrt{2D\Delta t}\nu_j^n, \quad n = 0, \ldots, N-1, \end{cases} \tag{3.7}$$

where $\Delta t = T/N$.

After all the simulations are carried out the approximate solution at T is obtained as a bar chart placing the particles in buckets and counting them. Let x_i be the centre of the i-th bucket and b_i the number of particles in it. The approximate solution in x_i is

$$u(x_i, T) = \frac{b_i}{J\Delta x},$$

where Δx is the width of the buckets (see Figure 3.1).

Notice that solving this particularly simple problem we could have used one-step walks: this will already recreate the solution since this is itself Gaussian.

The initial condition has been treated placing all the particles at $x = 0$. In general, initial conditions are to be interpreted as probabilities. The staring points of the trajectories are assigned randomly and a mechanism of rejection/reception is implemented to simulate the initial condition.

3.2.2 Finite domain

Consider the homogeneous Dirichlet boundary value problem

$$\begin{cases} U_t = DU_{xx}, \quad x \in (0,1),\ t \in (0,T], \\ U(x,0) = \delta(x) \\ U(0,t) = 0 = U(1,t). \end{cases} \tag{3.8}$$

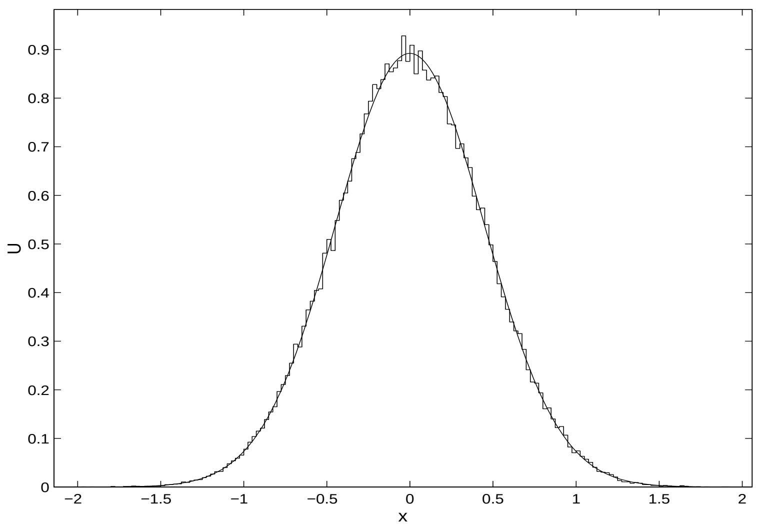

Figure 3.1: The analytic and the Monte Carlo solution of (3.6) for $T = 0.1$ after 100000 simulations.

The random walk is computed using $x = 0$ as starting point: for $j = 0, \ldots, J$

$$\begin{cases} X_j^0 = 0, \\ X_j^{n+1} = X_j^n + \sqrt{2D\Delta t}\,\nu_j^n, \quad n = 0, \ldots, N-1, \end{cases} \tag{3.9}$$

where, as usual, $\nu \sim N(0,1)$.

If at any time the particle hits the boundary its random walk is stopped and the particle discarded, as one would do in order to simulate a barrier in pricing an option.

On the contrary, if in the problem homogeneous Neumann boundary conditions are given, when a particle hits the boundary, the random walk is continued as if the particle bounced off a wall.

Figure 3.2 shows the solution represenented as a bar chart for $T = 0.1$.

3.2.3 Non-uniform diffusivity

Our last model problem, which we will discuss again in Section (2.3) where the adjoint method is presented, refers to equation (3.5). Consider

$$\begin{cases} \dfrac{\partial U}{\partial t} = \dfrac{\partial^2}{\partial x^2}\Big(D(x)U\Big), \quad x \in \mathcal{R},\, t > 0 \\ U(x,0) = \delta(x), \end{cases} \tag{3.10}$$

where

$$D(x) = \begin{cases} D^+, & \text{if}\, x > 0, \\ D^-, & \text{if}\, x < 0, \end{cases}$$

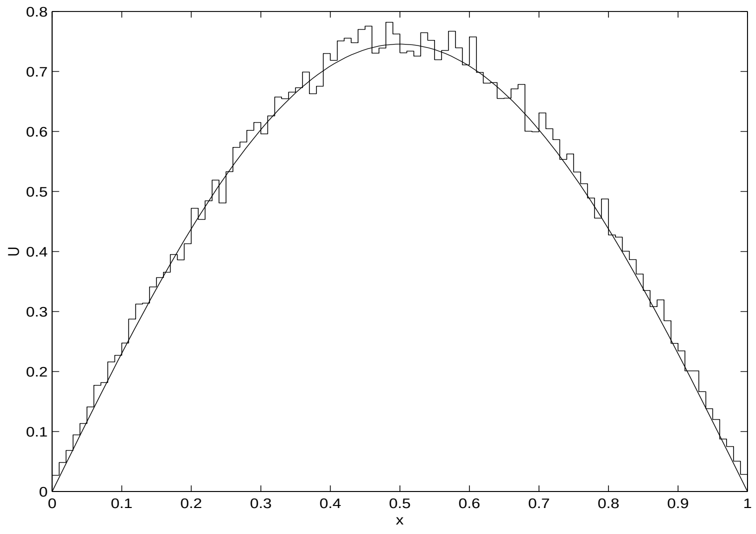

Figure 3.2: The analytic and the Monte Carlo solution of (3.8) with for $T = 0.1$ after 50000 simulations.

with $D^-, D^+ > 0$. The analytical solution of (3.10) can be found using Laplace Transforms. The Laplace Transform of U with respect to t is defined as

$$\overline{U}(x,s) = \int_0^{+\infty} e^{-st} U(x,t)dt.$$

Thus,

$$\overline{\left(\frac{\partial U}{\partial t}\right)} = -\delta(x) + s\overline{U}(x,s)$$

and in terms of $\overline{U}$ we have to solve the ordinary differential equation

$$s\,\overline{U}(x,s) - \delta(x) = \frac{d^2}{dx^2}(D\overline{U}(x,s)). \tag{3.11}$$

For $x > 0$ we define

$$\overline{U}^+(x,s) := \overline{U}(x,s),$$

and for $x < 0$ let

$$\overline{U}^-(x,s) := \overline{U}(x,s).$$

The equation satisfied by $\overline{U}^\pm(x,s)$ is

$$s\overline{U}^\pm = \frac{d^2}{dx^2}(D^\pm \overline{U}^\pm(x,s)).$$

Solving these equations, since DU is continuous, we obtain

$$\overline{U}^\pm = \frac{\overline{U}_0(s)}{D^\pm} e^{\mp\sqrt{\frac{s}{D^\pm}}x}.$$

Integrating (3.11) over the infinitesimal interval $(0^-, 0^+)$ gives

$$\begin{aligned} -1 &= \frac{d}{dx}(D^+\overline{U}^+)_{|x=0} - \frac{d}{dx}(D^-\overline{U}^-)_{|x=0} \\ &= -\sqrt{\frac{s}{D^+}}\overline{U}_0(s) - \sqrt{\frac{s}{D^-}}\overline{U}_0(s) \\ &= -\sqrt{s}\left(\frac{\sqrt{D^-}+\sqrt{D^+}}{\sqrt{D^-}\sqrt{D^+}}\right)\overline{U}_0(s). \end{aligned}$$

Hence

$$\overline{U}_0(s) = \frac{\sqrt{D^-}\sqrt{D^+}}{\left(\sqrt{D^-}+\sqrt{D^+}\right)\sqrt{s}}.$$

Thus, the solution to (3.11) is given by

$$\overline{U}^{\pm}(x,s) = \frac{\sqrt{D^{\mp}}}{\sqrt{D^{\pm}}\left(\sqrt{D^-}+\sqrt{D^+}\right)\sqrt{s}}\, e^{\mp\sqrt{\frac{s}{D^{\pm}}}x}.$$

Finally, applying the inverse Laplace transform we find the solution of the original

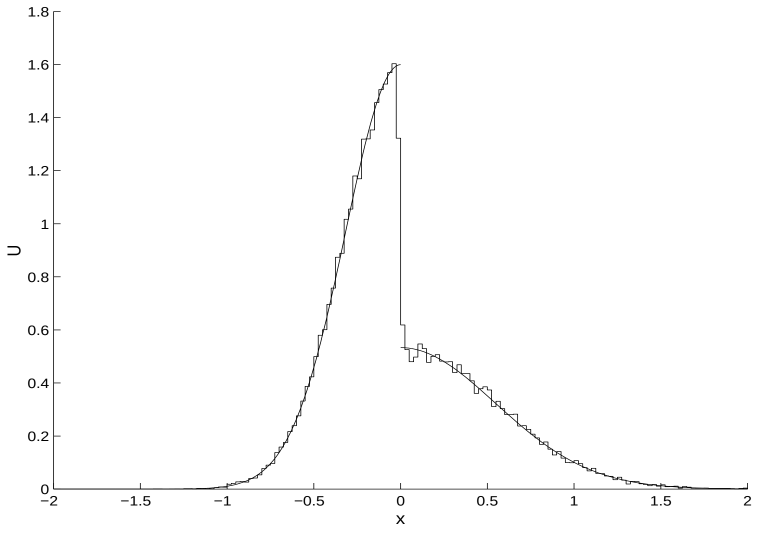

Figure 3.3: The analytic and the Monte Carlo solution of (3.10) for $T = 0.1$ after 50000 simulations.

problem

$$U^{\pm}(x,t) = \frac{\sqrt{D^{\mp}}}{\sqrt{D^{\pm}}\left(\sqrt{D^-}+\sqrt{D^+}\right)\sqrt{\pi t}}\, e^{-\frac{x^2}{4D^{\pm}t}}.$$

represented in Figure 3.3 together with a numerical solution for $t = 0.1$.

The solution is Gaussian for $x > 0$ and $x < 0$ and shows a discontinuity for $x = 0$ due to the discontinuity in the diffusivity (of course, for $D^+ = D^-$, we get (3.2)).

We used the analytical solution to check the convergence of our method (3.7). The method, as we see in Figure 3.4, is, as expected, of first order in Δt. For instance, the rate of convergence can be improved using a higher order Runge-Kutta method instead of (3.7).

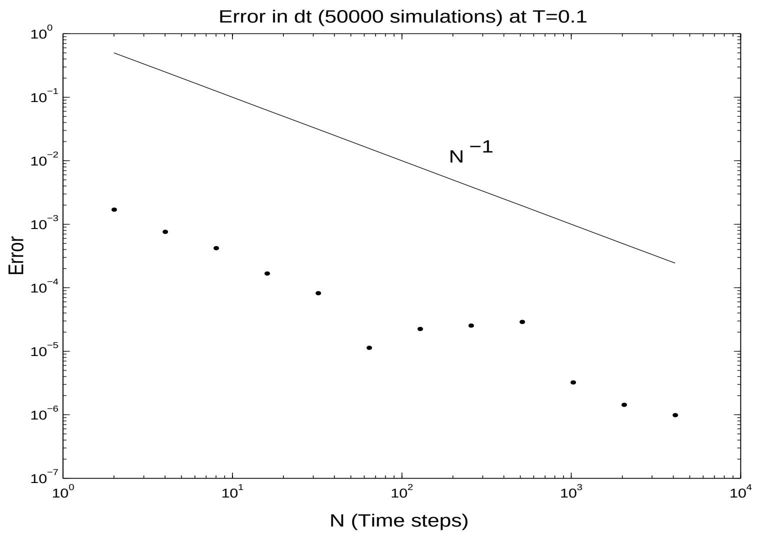

Figure 3.4: Error in the objective function due to time discretization.

The error represented is

$$error = (R(u^J) - R(U))^2 + (T(u^J) - T(U))^2 \tag{3.12}$$

where u^J is the Monte Carlo solution and R, and T are the second and third moment; thus

$$R(U) = \int x^2 U(x,T)dx,$$
$$T(U) = \int x^3 U(x,T)dx.$$

We have not used the first moment since this is zero: using the equation and integrating by parts

$$\begin{aligned} \frac{d}{dt}\left(\int_{-\infty}^{+\infty} xU(x,t)dx\right) &= \int_{-\infty}^{+\infty} x\frac{\partial U}{\partial t}(x,t)dx \\ &= \int_{-\infty}^{+\infty} x\frac{\partial^2}{\partial x^2}\left(D(x)U(x,t)\right)dx \\ &= x\frac{\partial}{\partial x}\left(D(x)U(x,t)\right)\Big|_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty}\frac{\partial}{\partial x}\left(D(x)U(x,t)\right)dx \\ &= -D(x)U(x,t)\Big|_{-\infty}^{+\infty} = 0 \end{aligned}$$

thus the statement since the first moment is zero for $t = 0$.

3.3 Discrete random walk

An alternative to applying Monte Carlo simulation to the solution of differential equations is to discretize the equation using a finite difference scheme.

For example, consider the case of constant diffusivity. Let u_i^n represent the solution on an equispaced grid with space-step Δx and time-step Δt and examine a finite difference scheme like the first order explicit scheme

$$u_i^{n+1} = (1 - 2\lambda)u_i^n + \lambda(u_{i+1}^n + u_{i-1}^n),$$

where $\lambda = D\Delta t/\Delta x^2$ with $\lambda \leq 1/2$ in order to satisfy the CFL condition.

The coefficients in the difference equation can be interpreted as probabilities that a particle reaches x_i at t_{n+1} starting from x_i, x_{i+1} and x_{i-1} respectively. This gives in a way the idea for the Monte Carlo approach to the solution of the difference equation.

The initial condition $f(x)$ is simulated by placing at each grid point $[Nf(x_i)]$ particles which will then follow a random walk through the time-space grid. At each step we generate a uniformly distributed number ν between 0 and 1 and

- if $0 < \nu \leq \lambda$, the particle moves on the left,
- if $\lambda < \nu \leq 2\lambda$, the particle moves on the right,
- if $2\lambda < \nu < 1$, the particle stays put.

This method is intuitive but it loses some of the good qualities of Monte Carlo; in particular, it requires a grids. Nevertheless, it can give us a hint of how to handle a problem where a stochastic interpretation is not readily available.

3.4 Linear sensitivities

An inverse problem is one in which the correct value of some parameter has to be found as function of the solution of a differential equation.

This leads to an optimisation problem for same objective function (least squares, etc.).

The objective of this thesis is to show that it is possible and advantageous to implement discrete adjoint methods to evaluate linear sensitivities. These are methods which have been already applied successfully, for example, to problems of optimal design where a large number of *design variables* is involved (see [14]). Our goal is to prove that they can be used in problems of implied volatility estimation.

In this chapter we present our technique on a model inverse problem of diffusivity estimation for the diffusion equation (3.10). We will assume that the diffusivity is unknown as function of some design variables and that it has to be evaluated minimising a scalar objective function.

The minimisation problem is treated numerically keeping fixed the random inputs. The approximate value of the minimum so obtained is expected to converge to the correct value of the diffusivity for $J \to \infty$ if the optimisation problem is well-posed.

The gradient of the objective function can be obtained with linear perturbation. The *linear sensitivity* in each design variable is obtained solving the linearised equation corresponding to a unit perturbation in a single design variable. From these we obtain the linear sensitivities for the objective function corresponding to the perturbed variable.

But we need to ensure that the dependence of the objective function on the design variables (same parametrisation of the diffusivity) is locally linear. This can be a problem when Monte Carlo simulation is the numerical method used since we compute only a finite number of trajectories. This is better understud with an example.

Suppose that we are given the integral $I(T)$ of the solution of (3.8) and that we are asked to deduce the correct constant diffusivity which will be the design variable. This inverse problem is well-posed since the integral is a monotonic function of the diffusivity. We have to solve a one dimensional unconstrained optimisation problem such as to minimise the objective function

$$obj = (I_J(D) - I(T))^2, \tag{3.13}$$

where $I_J(D)$ is the integral of the Monte Carlo solution which is given by

$$p(0 < X^N < 1).$$

In other words, the integral is given by the ratio between the number of particles that remaine in the domine and the total number of particles J which is kept fixed.

We simulate a finite number of particles, thus an infinitesimal perturbation in D results in a finite (or zero) change in the objective function. Hence the objective function is discontinuous in D. This is clearly visible in Figure 3.5 which shows the objective function evaluated as a function of D (using always the same random inputs). The dashed line shows the smoother objective function related to the modified optimisation problem which is to minimise

$$obj = \left(\frac{1}{J} \sum_j h(X_j^N) - \int_0^1 h(x)U(x)dx \right)^2 \tag{3.14}$$

where $h \in \mathcal{C}_c^0([0,1])$ is given.

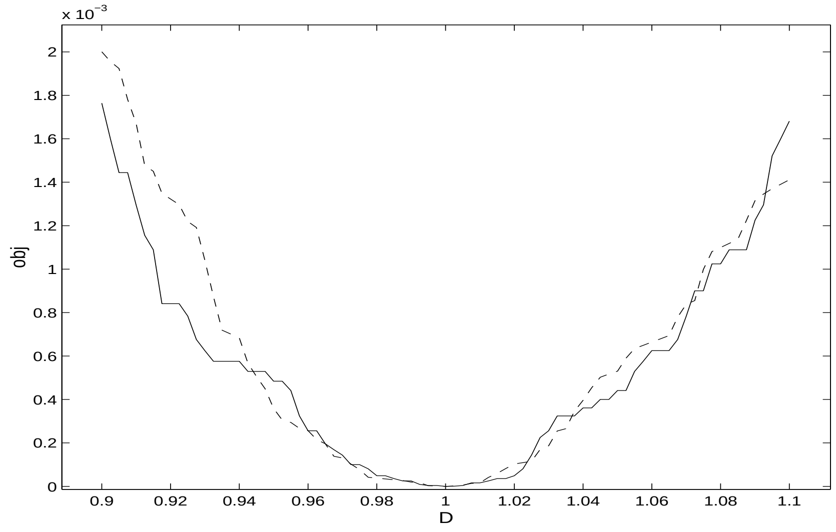

Figure 3.5: The objective functions (3.13) and (3.14) (dashed) in function of D. Both have been calculated keeping fixed all the parameters: the number of trajectories (and the random inputs) and the number of time steps involved.

For the new problem, the linear sensitivity at D_0 is obtained as follows.

We consider a perturbation in D_0. The gradient of the objective function is given by the value of the linear objective function

$$\widetilde{obj} = \frac{2}{J}\, obj^{1/2} \sum_j h'(X_j^N)\tilde{X}_j^N \tag{3.15}$$

where $\tilde{X}$ represents the solution of the equation linearised w.r.t. D. So to evaluate the gradient sensitivity we carry out simultaneously the random walk of the particle and its linearisation

$$\begin{cases} \tilde{X}_j^0 = 0, \\ \tilde{X}_j^{n+1} = \tilde{X}_j^n + \sqrt{2\Delta t}\frac{1}{2\sqrt{D}}\nu^n, \quad n = 0, \dots, N-1. \end{cases} \tag{3.16}$$

Figure 3.6 shows the objective function evaluated for some values for the diffusivity (the reference value being $D = 1$) and the gradient at $D_0 = 0.8$ found using (3.15).

We have checked our result comparing it to the derivative given by the complex variable method (see [14]). Given a function f, if f is analytic as complex function, then

$$f(x + ih) = f(x) + ihf'(x) + O(h^2),$$

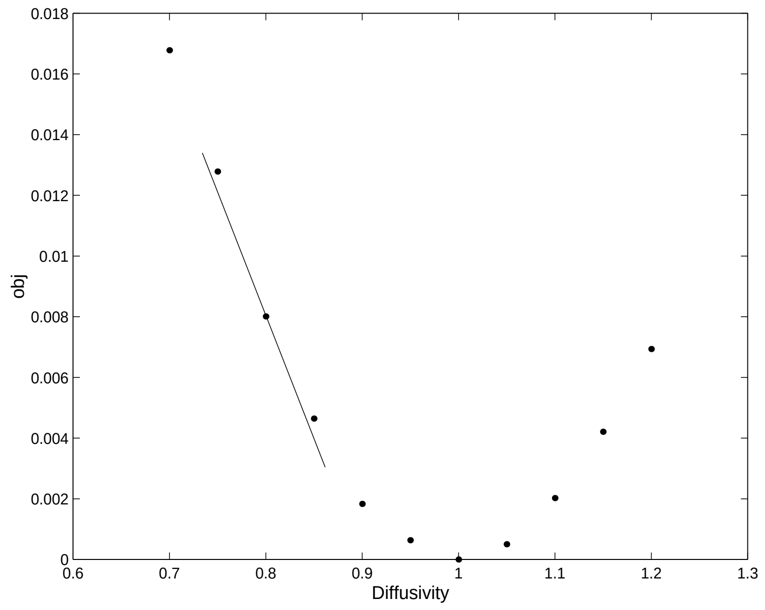

Figure 3.6: Linear sensitivity at $D = 0.8$.

hence

$$f'(x) \approx \frac{\mathcal{I}[f(x+ih)]}{h},$$

where the notation $\mathcal{I}$ denotes the imaginary part of a complex number. This formula is better then the usual finite difference approximation of the derivative since it is less sensitive to rounding errors. The reason being that when evaluated numerically with $h \ll 1$ it does not involves subtraction of two almost equal quantities.

Thus we can obtain the linear sensitivity defining

$$D = D_0 + ih,$$

and carrying out the simulations as in (3.9). The linear sensitivity will be given by

$$\widetilde{obj} = \frac{\mathcal{I}(obj)}{h}.$$

A third way to evaluate the linear sensitivity is through a discrete adjoint method. This is discussed in the next section on a model problem similar to (3.10), so with diffusivity varying in x, since the adjoint approach has more significance and so is better understood when more then one design parameter is involved.

3.5 The adjoint approach

We consider the problem of identifying the diffusion coefficient in

$$\begin{cases} \frac{\partial}{\partial t}U(x,t) = \frac{\partial^2}{\partial x^2}(D(x)U(x,t)), \quad x \in \mathcal{R}, \quad t \in [0,T] \\ U(x,0) = \delta(x), \quad x \in \mathcal{R} \end{cases} \tag{3.17}$$

when the dependence of D on x is trough the value of two unknown parameters.

We assume that the diffusivity passes smoothly from the (constant) value D^- to D^+ (our design variables) obeying a low as

$$D(x) = D^- d^-(x) + D^+ d^+(x).$$

This is to avoid the problem discussed in the previous section. For instance define

$$d^{\pm}(x) := \frac{1}{2}(1 \pm \tanh(\frac{2\pi}{a}x))$$

for $a > 0$ so that $D(x) = D^-$ for $x < -a$ and $D(x) = D^+$ for $x > a$ with D^-, $D^+ > 0$.

As usual, the Monte Carlo approach is used to compute a large number J of discrete random walks starting from $x = 0$. Thus, for $j = 1, \ldots, J$ we simulate

$$\begin{cases} X_j^0 = 0, \\ X_j^{n+1} = X_j^n + \sqrt{2\Delta t D(X_j^n)}\, \nu_j^n, \quad n = 0, \ldots, N-1, \end{cases} \tag{3.18}$$

where $\nu \sim N(0,1)$.

Let us define the discrete standard deviation of the final positions of the trajectories

$$R(u) = \sqrt{\frac{1}{J}\sum_j (X_j^N - \bar{X}^N)^2},$$

where $\bar{X}^N$ represents the mean, and the third moment

$$T(u) = \frac{1}{J}\sum_j (X_j^N)^3.$$

The parameter identification is archived minimising the least squares objective functional

$$obj = \big(R(u) - R(u_r)\big)^2 + \big(T(u) - T(u_r)\big)^2,$$

where u_r is a solution given by a reference calculation. Figure 3.7 shows the objective functional in function of D^- and D^+.

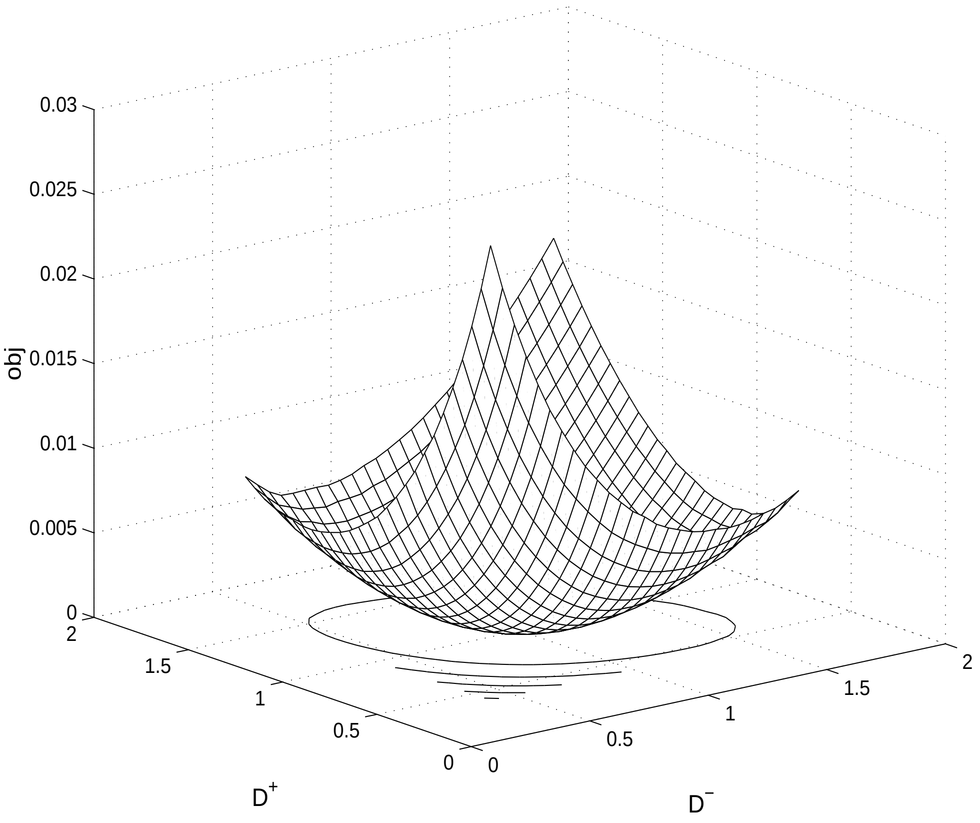

Figure 3.7: The least squares functional with reference parameters fixed at $D^{\pm} = 1$.

To get the gradients we perform the linear perturbation analysis. In the 'direct' mode, this is done computing simultaneously the trajectories (3.18) and their linearisation with respect to the variable that is perturbed, that is for $j = 1, \dots, J$

$$\begin{cases} \widetilde{X}_j^0 = 0, \\ \widetilde{X}_j^{n+1} = \tilde{X}_j^n + \sqrt{2\Delta t}\Big(\frac{1}{2}\frac{\widetilde{D(X_j^n)}}{\sqrt{D(X_j^n)}} + \frac{1}{2}\frac{D'(X_j^n)}{\sqrt{D(X_j^n)}}\widetilde{X}_j^n\Big)\nu_j^n, \quad n = 1\dots, N-1, \end{cases} \tag{3.19}$$

where

$$\tilde{D} = \begin{cases} d^-, & \text{for perturbation in } D^-, \\ d^+, & \text{for perturbation in } D^+, \end{cases}$$

and D' represents derivative in X.

The gradient of the objective function on one design variable is given by the value of the linearised objective function

$$\begin{aligned}
\widetilde{obj}^{\pm} &= \frac{\partial}{\partial D}\Big(\big(R(u)-R(u_r)\big)^2+\big(T(u)-T(u_r)\big)^2\Big) \\
&= \frac{2}{J}\sum_{j=1}^{J}\Big[\Big(1-\frac{R(u_r)}{R(u)}\Big)(\alpha_j-\bar{\alpha})+\frac{3}{J}(T(u)-T(u_r))\sum_{j}(X_j^N)^2\Big]\tilde{X}_j^N \\
&=: \sum_{j=1}^{J} f_j(X_1^N,\dots,X_J^N)\tilde{X}_j^N, \qquad (3.20)
\end{aligned}$$

where

$$\alpha_j = X_j^N - \bar{X}_J^N,$$
$$\bar{\alpha} = \frac{1}{J}\sum_{j=1}^{J}\alpha_j$$

This procedure is reformulated in the context of linear algebra as follows. For every simulation let

$$A_j = \begin{pmatrix} 1 & & & \\ -1-\frac{\sqrt{2\Delta t}D'(X_j^1)}{2\sqrt{D(X_j^1)}}\nu_j^1 & 1 & & \\ & \ddots & \ddots & \\ & & -1-\frac{\sqrt{2\Delta t}D'(X_j^N)}{2\sqrt{D(X_j^N)}}\nu_j^N & 1 \end{pmatrix}$$

$$b_j^{\pm} = \begin{pmatrix} 0 \\ \frac{\sqrt{2\Delta t}\tilde{D}^{\pm}(X_j^N)}{2\sqrt{D(X_j^N)}}\nu_j^1 \\ \vdots \\ \frac{\sqrt{2\Delta t}\tilde{D}^{\pm}}{2\sqrt{D}}\nu_j^N \end{pmatrix}, \quad g_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ f_j(X_1^N,\dots,X_J^N) \end{pmatrix}.$$

The computation of (3.19) is equivalent to the solution by forward substitution of the bidiagonal linear system $A_j\tilde{X}_j^{\pm}=b_j^{\pm}$.

So, to get the gradient sensitivities we

- SOLVE $A_j\tilde{X}_j^{\pm}=b_j^{\pm}$, for $j=1,\dots,J$

- EVALUATE $\sum_j g_j^T \cdot \tilde{X}_j^{\pm}$.

An alternative way to do this is via the 'discrete' adjoint approach, which gives the gradient sensitivities trough the dual formulation

- SOLVE $A_j^T Y_j^N = g_j$, for $j = 1, \ldots, J$
- EVALUATE $\sum_j (Y_j^N)^T \cdot b_j^{\pm}$.

The equivalence of the two approaches is easily proved

$$g_j^T \cdot \tilde{X}_j = g_j^T A_j^{-1} b_j = \left((A_j^T)^{-1} g_j\right)^T b_j = Y_j^T \cdot b.$$

So, mathematically, the adjoint approach is equivalent to obtaining linear sensitivities.

In general, the adjoint approach can be cheaper if we need to evaluate $g^T \cdot \tilde{X}$ for m different values of g and p different values of f and $m \ll p$, because it involves the solution of a smaller number of linear systems.

In the simple model problem for the heat equation under discussion, we have two parameters so we have to deal with two values for b but only one value for g.

The choice will be either to

- solve $A_j \tilde{X}_j^{\pm} = b_j^{\pm}$, which is done by forward substitution as in (3.19)
- evaluate $g_j^T \cdot \tilde{X}_j^{\pm}$,

or to proceed on the dual form

- solve $A_j^T Y_j = g_j$ which is done by backward substitution:

$$\begin{aligned} Y_j^N &= f_j \\ Y_j^{n-1} &= Y_j^n + \frac{\sqrt{2\Delta t} D'(X_j^n)}{2\sqrt{D(X_j^n)}} Y_j^n \nu_j^n, \quad n = N, \ldots, 1 \end{aligned}$$

- evaluate $(Y_j^N)^T \cdot b^{\pm}$.

Figure 3.8 shows that the adjoint method gives the correct value for the gradient (in this case in D^+). As expected, we found that the adjoint method gives the same gradients obtained with the direct method apart for rounding errors.

Notice that the numerical minimum (which depends on J) is far from the correct one which is zero and is taken at $D^{\pm} = 1$ in our case.

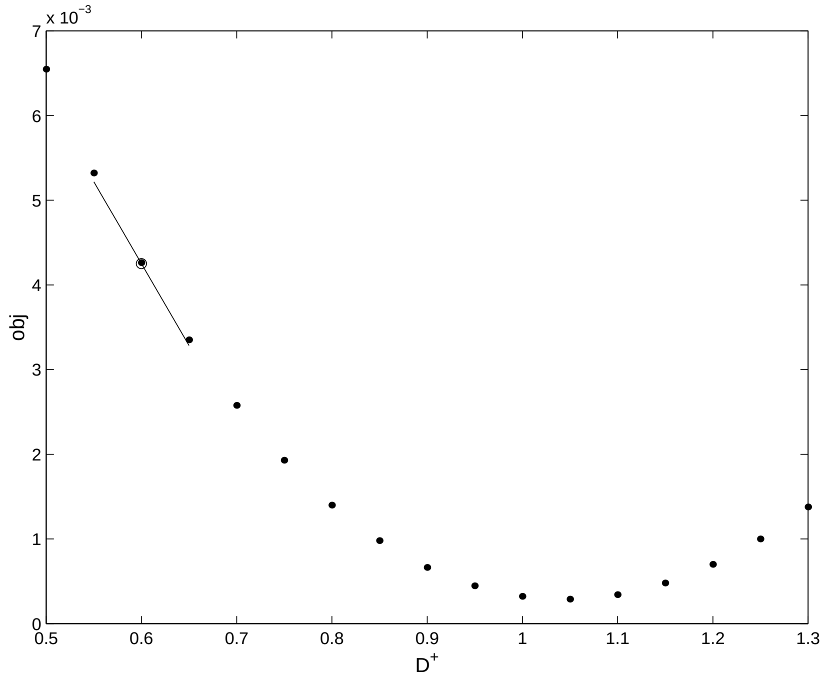

Figure 3.8: The objective function plotted against D^+ with D^- kept fixed and the linear sensitivity at $D = 0.6$ evaluated using the discrete adjoint method.

3.6 Practical issues

The linearised objective functional depends through f_j on the standard deviation and on the third moment of the final positions of all the trajectories.

Thus, in order to be able to start the backward substitution, we need to simulate all the trajectories storing the final positions and then evaluate f_j for every j.

Moreover, the nonlinear term in D of (4.9) depends on the whole trajectory X_j^n. So, since all the trajectories need to be calculated before we start the backward substitution, we actually need to memorise all the steps of all the trajectories.

This is really too expensive in terms of memory usage. So, as it is, the algorithm is impracticable.

What we actually do is the following:

- We compute all the forward calculations once and evaluate f_j from the final positions.
- In a new do loop, we compute every trajectory once again. Once that one trajectory is computed storing all the intermediate steps, the related backward substitution can be calculated.

- The linear sensitivities are obtained summing the contribution of all the trajectories.

Notice that in this way we need only to store one trajectory at a time. The draw back is that the trajectories have to be simulated twice. So, as far as only few design parameters are involved, the adjoint approach will not be cheaper in terms of computational work. We will discuss this point again later.

Chapter 4

Adjoint methods and volatility surfaces

4.1 Model problem and linear sensitivities

Suppose that we are given a finite number of market prices for vanilla call options all with the same expiry but with different strike prices and that we have to find the implied volatility in function of S. To handle such a problem, we choose a parametrisation for the volatility and we try to minimise the difference between the given prices and the Black-Scholes prices.

For simplicity, suppose that two prices V^1 and V^2 are given for two different strikes E^1 and E^2. As far as the volatility is concerned, we suppose that it depends continuously on two parameters σ^+ and σ^- and we choose the representation

$$\sigma(S) = \sigma^- d^-(S) + \sigma^+ d^+(S), \tag{4.1}$$

where, as in Section (2.5),

$$d^{\pm}(S) = \frac{1}{2}(1 \pm \tanh(\frac{2\pi}{a}(S - S_0)).$$

Here S_0 represents a reference price (today's price in our implementations).

This representation is quite arbitrary but it gives us a good model problem for which we will prove that it is possible to implement the discrete adjoint method to evaluate linear sensitivities. It is straight-forward to apply the method to different representations.

The optimisation problem is to minimise the least square objective functional

$$obj = \sum_{i=1}^{2} \left(V_J^i(\sigma^-, \sigma^+) - V^i\right)^2, \tag{4.2}$$

where V_J^i is the Monte Carlo estimation of today's value of the option (J and the trajectories are kept fixed while the optimisation is carried on).

Thus, assuming that the risk-free rate is constant,

$$V_J^i = \frac{e^{-r(T-t)}}{J} \sum_{j=1}^{J} \max(S_j^N - E^i, 0), \tag{4.3}$$

where the final prices S_j^N are obtained simulating the asset price using (4.1) as volatility:

$$\begin{cases} S_j^0 = S(t), \\ S_j^{n+1} = S_j^n + rS_j^n \Delta t + \sigma(S_j^n) S_j^n \sqrt{\Delta t}\, \nu_j^n, \quad n = 0, \ldots, N-1, \end{cases} \tag{4.4}$$

where $\nu \sim N(0,1)$.

As usual, the linear sensitivities are given by the linearisation of the objective functional with respect to the parameter being perturbed.

Deriving (4.4) with respect to D^+ or D^- we have

$$\begin{cases} \tilde{S}_j^0 = 0, \\ \tilde{S}_j^{n+1} = \tilde{S}_j^n (1 + r\Delta t + \sigma(S_j^n)\sqrt{\Delta t}\, \nu_j^n) + S_j^n \left(\tilde{\sigma}(S_j^n) + \sigma'(S_j^n)\tilde{S}_j^n \right) \sqrt{\Delta t}\, \nu_j^n, \end{cases} \tag{4.5}$$

where

$$\tilde{\sigma}(S) = \begin{cases} d^+(S), & \text{for derivation w.r.t. } D^+, \\ d^-(S), & \text{for derivation w.r.t. } D^- \end{cases}$$

and σ' represents the derivative of σ with respect to S, so

$$\begin{aligned} \sigma'(S) &= \sigma^- d'_-(S) + \sigma^+ d'_+(S) \\ &= \frac{\pi}{a}(\sigma^+ - \sigma^-) \operatorname{sech}^2 \left(\frac{2\pi}{a}(S - S_0) \right) \\ &= \frac{\pi}{a}(\sigma^+ - \sigma^-) \left(1 - \tanh^2 \left(\frac{2\pi}{a}(S - S_0) \right) \right). \end{aligned}$$

Let J_*^1, J_*^2 represent the indices for the trajectories which ends in the money, i.e. for which $X_j^N - E^1 > 0$ and $X_j^N - E^1 > 0$ respectively.

The linearised objective functional is

$$\widetilde{obj}^{\pm} = 2\, \frac{e^{-r(T-t)}}{J} \Big[\big(V_J^1(\sigma^-, \sigma^+) - V^1 \big) \sum_{j_*^1} \tilde{S}_{j_*^1}^N + \big(V_J^2(\sigma^-, \sigma^+) - V^2 \big) \sum_{j_*^2} \tilde{S}_{j_*^2}^N \Big] \tag{4.6}$$

Our final goal is to evaluate the linearised objective function (4.6). To do that we proceed as follows:

- S, $\tilde{S}^+$ and $\tilde{S}^-$ are simulated simultaneously,

- as a trajectory is carried over, if this ends in the money, the payoffs and the values $\tilde{S}_j^N$ are summed to the quantities already calculated,

- as all the trajectories have been carried over, the option prices V^1, V^2 are evaluated as the mean of the discounted payoffs and obj, $\widetilde{obj}^+$ and $\widetilde{obj}^-$ are obtained.

4.2 The adjoint approach

We can write (4.6) as

$$\widetilde{obj}^{\pm} = \sum_{j_*} g \cdot \tilde{S}_{j_*}, \tag{4.7}$$

where, as in Section (2.5), g has only the last entry non zero. The computation of the linear perturbations is equivalent to the solution by forward substitution of the linear system

$$A_j \tilde{X}_j^{\pm} = b_j^{\pm}, \tag{4.8}$$

where A_j is a bidiagonal matrix defined by

$$\begin{aligned} A_j(n,n) &= 1, \quad n = 0, \ldots, N \\ A_j(n+1,n) &= -\left(1 + r\delta t + \sigma(S_j^n)\sqrt{\delta t}\,\nu_j^n + S_j^n \sigma'(S_j^n)\sqrt{\delta t}\,\nu_j^n\right), \quad n = 1, \ldots, N. \end{aligned}$$

and

$$b_j^{\pm} = \begin{pmatrix} 0 \\ S_j^1 \tilde{\sigma}^{\pm}(S_j^1)\sqrt{\Delta t}, \nu_j^1 \\ \vdots \\ S_j^N \tilde{\sigma}^{\pm}(S_j^N)\sqrt{\Delta t}, \nu_j^N \end{pmatrix}.$$

Thus, the linear sensitivities are evaluated solving (4.8) and then calculating the vector dot products of (4.7).

Alternatively, we proceed on the dual formulation:

- solve $A_j^T R_j = g$ which is done by backward substitution:

$$\begin{aligned} R_j^N &= g \\ R_j^{n-1} &= \left(1 + r\delta t + \sigma(S_j^n)\sqrt{\delta t}\,\nu_j^n + S_j^n \sigma'(S_j^n)\sqrt{\delta t}\,\nu_j^n\right) R_j^n, \\ & \qquad n = N, \ldots, 1. \end{aligned} \tag{4.9}$$

- evaluate $R_j^T \cdot b_j^{\pm}$.

This is the adjoint approach for the evaluation of linear sensitivities which we have already proved to be equivalent to the direct approach (see Section 3.6).

To save memory, all the trajectories are computed a first time and the objective function is obtained. Then, every trajectory is computed ones again together with the backward substitution and the contributions to the linearised objective function are summed up.

We have computed the linear sensitivities applying both the direct and the adjoint approach and we have found that the two approaches gives the same results apart from rounding errors. The objective function is plotted together with the gradient in Figure 4.1 in function of one of the two design parameters.

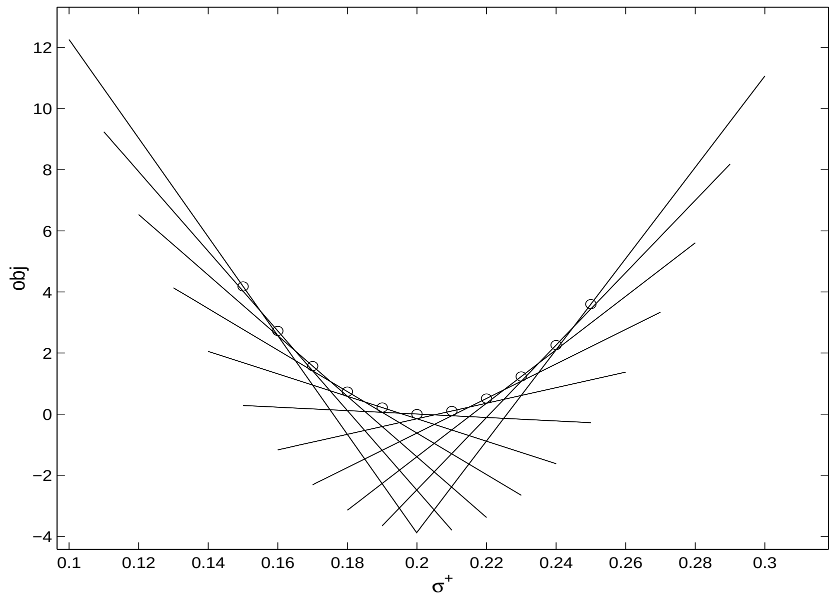

Figure 4.1: The objective function and its gradient for different values of σ^+.

From the point of view of the computational work, since the trajectories are calculated twice, the adjoint approach is cheaper only when a large number of design parameters are involved.

Even in the letter case, it could be still not clear why there should be a gain, since the linear systems we deal with are solvable by forward substitution. In reality, if, as it happens in most applications, the design variables have compact support, then the b_j's are almost zero. Hence, an adjoint algorithm which makes use of this characteristic is faster then a direct calculation.

The programming language we used to test the algorithms is MATLAB. This simplifies the programming of all the surrounding calculations.

We are aware that this is not the optimal tool to use to implement Monte Carlo simulation (FORTRAN and C++ are the languages commonly used). Nevertheless, MATLAB is very fast in working with vectors. In writing our codes we have used this characteristic by simulating many trajectories at a time. This loweres the computational time drastically making of MATLAB at least a good alternative.

4.3 Conclusions

We have proved the validity of the adjoined approach for linear sensitivities evaluation in the contest of deterministic implied volatility surfaces. The method should be tested on more complicated volatility representations to ascertain the computational advantages. In particular, we would like to test the method with real market datas and on implied volatility surface representations used in practice. For example, many options are priced everyday using piecewise linear representations for the volatility. Finally, an exhaustive test of the method should include an algorithm for the actual solution of the optimisation problem.

Appendix A

MATLAB codes

A.1 Linear sensitivities for the non-uniform diffusivity problem

```
% PROGRAM 1 %
% Linear sensitivities using the adjoint approach %
% for the problem of Section 3.5 %

clear
% final time %
tau = 0.1;
% reference calculation %
D_P = 1; D_M = 1;
N_t = 1000; no_sim_t = 10000;
dt = tau/N_t;
dx = sqrt(2*dt);
for j=1:no_sim_t
  x = 0;
  ran = randn(1,N_t);
  for n=1:N_t
    effe=tanh(4*pi*x)/2;
    x = x + dx * sqrt(D_P*(.5+effe)+D_M*(.5-effe)) * ran(n);
  end
  ref(j) = x;
end
rms_t = std(ref);
```

```
third_t = mean(ref.^3);

% simulation parameters %
N = 200;
no_sim = input('Number of simulations ');
% values of the design variables at %
% which the gradient of obj is evaluated %
D_P = .6;
D_M = 1;
% forward calculation of the trajectories %
dt = tau/N; dx = sqrt(2*dt);
difstep = sqrt(2*dt);
good = zeros(1,no_sim);
for j=1:no_sim
  s(:,j) = randn('state'); ran = randn(1,N);
  x = 0;
  for n=1:N
    effe=tanh(4*pi*x)/2;
    x = x + dx * sqrt(D_P*(.5+effe)+D_M*(.5-effe)) * ran(n);
  end
  good(j) = x;
end
rms = std(good); third = mean(good.^3);
obj = (third-third_t)^2+(rms-rms_t)^2; % objective function %
f=2*( (3*(third-third_t)*good.^2)/no_sim+(1-rms_t/rms)...
   *(good-media-mean(good-mean(good)))/(no_sim-1) );

% linear sensitivities (adjoint)
b_P = zeros(1,N+1);  %% it is important that b(1)=0 %%
b_M = zeros(1,N+1);
objta_P=0; objta_M=0;
for j=1:no_sim
  randn('state',s(:,j)); ran = randn(1,N);
  x = 0; v = zeros(1,N+1);
  v(N+1) =  f(j);
  for n=1:N
```

```
    effe = tanh(4*pi*x)/2;
    dprime = 2*pi*(1-(tanh(4*pi*x))^2);
    dprv(n) = dprime;
    sqrt_D = sqrt(D_P*(.5+effe)+D_M*(.5-effe));
    sqrtdv(n) = sqrt_D;
    x = x + dx * sqrt_D * ran(n); % trajectories %
    b_P(n+1) = .5 * dx * (.5+effe) *ran(n)/sqrt_D; % RHS %
    b_M(n+1) = .5 * dx * (.5-effe) *ran(n)/sqrt_D;
  end
  for n=N:-1:1
    v(n) = v(n+1) + .5 * dx * (D_P*dprv(n)-D_M*dprv(n))...
           * ran(n) * v(n+1)/sqrtdv(n);
  end
  objta_P = objta_P + sum(v.*b_P);
  objta_M = objta_M + sum(v.*b_M);
end
objta_P, objta_M % linear sensitivities %
```

A.2 Evaluation of vanilla call prices

```
% PROGRAM 2 %
% Monte Carlo price of a call option using %
% control variate and %
% antithetic variables %

clear
% problem parameters %
rate = .05; drift = .05;
vol = .2; strike = 105;
asset = 100; T = 1; t=0;
discount = exp(-rate*(T-t));
% control variate %
cv_strike=102;
cv_value=bscall(asset,rate,vol,T,t,cv_strike);
% simulation parameters %
no_sim = 100000; % Number of simulations (multiple of simdim) %
simdim=1000; % Number of simulations computed simultaneously %
loop = no_sim/simdim;
N=round((no_sim)^(1/2));
dt = (T-t)/N
% simulation loop %
sq_vol = vol*vol; sqrt_dt = sqrt(dt); c = 0; cv=0;
for i=1:loop
  x1 = ones(1,simdim)*asset;
  x2 = ones(1,simdim)*asset; % antithetic variable %
  for n=1:N
    ran = randn(1,simdim);
    x1 = x1.*exp((rate-0.5*sq_vol)*dt+vol*sqrt_dt*ran);
    x2 = x2.*exp((rate-0.5*sq_vol)*dt-vol*sqrt_dt*ran);
  end
% payoffs discounted %
  c = c+sum(discount*max(x1-strike,0));
  c = c+sum(discount*max(x2-strike,0));
  cv = cv+sum(discount*max(x1-cv_strike,0)); % control variate %
  cv = cv+sum(discount*max(x2-cv_strike,0));
end
% call current value = mean of today value of the payoffs %
  call_val=c/(2*no_sim)-cv/(2*no_sim)+cv_value
```

```
% evaluation of the analytic call price

function call=bscall(S,r,vol,T,t,Strike)
d1=(log(S/Strike)+(r+0.5*vol*vol)*(T-t))/(vol*sqrt(T-t));
d2=(log(S/Strike)+(r-0.5*vol*vol)*(T-t))/(vol*sqrt(T-t));
N1=0.5*erfc(-d1/sqrt(2));
N2=0.5*erfc(-d2/sqrt(2));
call=S*N1-Strike*exp(-r*(T-t))*N2;
```

A.3 Linear sensitivities for implied volatility estimation

```
% Program 3
% Linear sensitivities (direct and adjoint approach) %
% for the problem of Section (4.1) %

clear
% problem parameters
rate = .05; drift = .05; asset = 100;
T = 1; t=0; strike_a = 107; strike_b = 102;
discount = exp(-rate*(T-t));
% simulation parameters
N = 200; % Number of time steps %
dt = (T-t)/N;
no_sim = 10000; % Number of simulations (multiple of simdim) %
simdim=1000; % Number of simulations computed simultaneously %
loop = no_sim/simdim;
% reference call values %
val_a = 8; val_b = 9;
% trajectories and direct linear sensitivities at vol1, vol2 %
sqrt_dt = sqrt(dt);
vol1 = .15; vol2 = .15;
c_a = 0; c_b = 0; ct1_a = 0; ct2_a = 0; ct1_b = 0; ct2_b = 0;
% simulation loop %
for j=1:loop
  x = ones(1,simdim)*asset;
  xt1 = zeros(1,simdim); xt2 = zeros(1,simdim);
  for n=1:N
    s(:,j,n) = randn('state'); ran = randn(1,simdim);
    effe=tanh(2*pi*(x-asset))/2;
    dprime = pi*(1-(tanh(2*pi*(x-asset))).^2);
    vol = (vol1*(.5+effe)+vol2*(.5-effe));
    dvol = (vol1*dprime-vol2*dprime);
% linear sensitivities: direct calculation %
    temp= (1+rate*dt+vol*sqrt_dt.*ran+x*sqrt_dt.*ran.*dvol;
    xt1 = temp.*xt1+x*sqrt_dt.*ran.*(.5+effe); % linear sensitiv.: %
    xt2 = temp.*xt2+x*sqrt_dt.*ran.*(.5-effe); %  direct calculation %
    x = x.*(1+rate*dt+vol*sqrt_dt.*ran); %  trajectories %
  end
% payoffs %
  payoff_a = max(x-strike_a,0); payoff_b = max(x-strike_b,0);
  c_a = c_a+sum(payoff_a); c_b = c_b+sum(payoff_b);
  for jj=1:simdim
    if (payoff_a(jj)~=0)
      ct1_a = ct1_a+xt1(jj); ct2_a = ct2_a+xt2(jj);
    end
    if (payoff_b(jj)~=0)
      ct1_b = ct1_b+xt1(jj); ct2_b = ct2_b+xt2(jj);
    end
  end
end
% call current value = mean of payoffs discounted %
call_value_a = (c_a*discount)/no_sim, call_value_b = (c_b*discount)/no_sim
g_a=2*(discount/no_sim)*(call_value_a-val_a);
g_b=2*(discount/no_sim)*(call_value_b-val_b);
% direct linear sensitivities %
lin_sens1 = g_a*ct1_a+g_b*ct1_b, lin_sens2 = g_a*ct2_a+g_b*ct2_b

% adjoint calculation %
b1 = zeros(N+1,simdim); b2 = zeros(N+1,simdim);
lin_sens_adj1 = 0; lin_sens_adj2 = 0;
% trajectories and linear sensitivities (adjoint)
for j=1:loop
  x = ones(N+1,simdim)*asset;
  v_a = zeros(N+1,simdim); v_b = zeros(N+1,simdim);
  v_a(N+1,:) =  g_a; v_b(N+1,:) =  g_b;
  for n=1:N
    randn('state',s(:,j,n));
    ran = randn(1,simdim);
    keep_ran(n,:) = ran;
    effe(n,:)=tanh(2*pi*(x(n,:)-asset))/2;
    dprime(n,:) = pi*(1-(tanh(2*pi*(x(n,:)-asset))).^2);
    vol(n,:) = (vol1*(.5+effe(n,:))+vol2*(.5-effe(n,:)));
    dvol(n,:) = (vol1*dprime(n,:)-vol2*dprime(n,:));
    b1(n+1,:) = sqrt_dt*(.5+effe(n,:))*(x(n,:)*ran(:)); % RHS %
    b2(n+1,:) = sqrt_dt*(.5-effe(n,:))*(x(n,:)*ran(:));
    x(n+1,:) = x(n,:)*(1+rate*dt+vol(n,:)*sqrt_dt*ran(:)); % trajectories %
  end
% payoffs %
  payoff_a = max(x-strike_a,0); payoff_b = max(x-strike_b,0);
  for jj=1:simdim
    if (payoff_a(jj)~=0)
      for n=N:-1:1
        v_a(n,jj) = (1+rate*dt+vol(n,jj)*sqrt_dt*keep_ran(n,jj)+...
        x(n,jj)*sqrt_dt*dvol(n,jj)*keep_ran(n,jj))*v_a(n+1,jj);
      end
    lin_sens_adj1 = lin_sens_adj1 + sum(v_a(:,jj).*b1(:,jj));
    lin_sens_adj2 = lin_sens_adj2 + sum(v_a(:,jj).*b2(:,jj));
    end
    if (payoff_b(jj)~=0)
      for n=N:-1:1
        v_b(n,jj) = (1+rate*dt+vol(n,jj)*sqrt_dt*keep_ran(n,jj)+...
        x(n,jj)*sqrt_dt*dvol(n,jj)*keep_ran(n,jj))*v_b(n+1,jj);
      end
    lin_sens_adj1 = lin_sens_adj1 + sum(v_b(:,jj).*b1(:,jj));
    lin_sens_adj2 = lin_sens_adj2 + sum(v_b(:,jj).*b2(:,jj));
    end
  end
end
lin_sens_adj1, lin_sens_adj2
```

Bibliography

[1] P. P. Boyle. Options: a monte carlo approach. *Journal of Financial Economics*, 4:323–338, 1977.

[2] R. E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, pages 1–49, 1998.

[3] J. M. Hammersley, D. C. Handscomb. *Monte Carlo Methods.* Chapman and Hall, London, 1964.

[4] E. Derman and I. Kani. Riding on a smile. *RISK Magazine*, 7 No. 2, February 1994.

[5] B. Dupire. Pricing with a smile. *RISK Magazine*, 7 No. 2, January 1994.

[6] W. Feller. *An Introduction to ProbabilityTheory and its Applications: Vol. I.* Wiley, 1971.

[7] J. Hull. *Options, Futures and Other Derivative Securities.* Prentice-Hall, 1993.

[8] P. Wilmott, S. Howison, J. Dewinne. *The Mathematics of Financial Derivatives.* CUP, 1995.

[9] N. A. Jackson. *Adaptive Finite Element Solution of Option Pricing Problems.* DPhil Thesis, Oxford University Computing Laboratory, 1999.

[10] W. J. Morokoff and R. E. Caflisch. A quasi-monte carlo approach to particle simulation of the heat equation. *SIAM J. Numer. Anal.*, 30 No. 2:1558–1573, December 1993.

[11] A. Okubo. *Diffusion and Ecological Problems: Mathematical Models.* Springer-Verlag, 1980.

[12] M. Broadie P. P. Boyle and P. Glasserman. Monte carlo methods for security pricing. *Monte Carlo: Methodologies and Applications for pricing and Risk Management*, Ed. B. Dupire.

[13] M. H. Kalos, P. Whitlock. *Monte Carlo Methods I: Basics.* Wiley, New York, 1986.

[14] N. A. Pierce and M. B. Giles. An introduction to the adjoint approach to design. *European Journal of Flow, Turbulence and Combustion*, 2000 (to appear).

[15] D. Schimko. Bounds on probability. *RISK Magazine*, 6(4):59–66, 1993.

[16] E. P. Gill, M. H. Wright, W. Murray. *Practical Optimisation.* Academic Press, 1981.

[17] Y. K. Kwok. *Mathematical Models of Financial Derivatives.* Springer, 1998.