Malliavin Calculus for Monte Carlo Simulation with Financial Applications

Eric Alexander Schiller

Advised by K.R. Sircar

Department of Mathematics
Princeton University

May 4, 2009

PRINCETON
UNIVERSITY

Preface

This work represents the practical application of abstract mathematics. In my time as a mathematician, I have never known how to feel about the applied side of the field. When I entered Princeton University, I thought that useful things were generally trivial and that engineers and scientists were fools to seek truth in data sets. My studies have led me to know I was wrong, and I now hold in the highest esteem those who use complicated ideas to solve recognized problems. Of all things I have gained at Princeton, perspective like this is dearest.

I am glad to have the opportunity to recognize those who have given me this perspective in general and those who have helped me write this work in particular. I am grateful to the dedicated teachers who have introduced me to the world of math: Deborah Foley, Ramin Takloo-Bighash, Bob Gunning, Jake Rasmussen, Robert Calderbank, Erhan Çinlar, and Eli Stein. But for their enthusiasm, I would not have found my way here. My advisor, Ronnie Sircar, is very serious about his study, and meeting with him regularly kept me working towards new discoveries and—more importantly—consolidating the ones I had made into presentable form. His guidance in this project, from suggesting the Malliavin calculus as an object of study to discussing my drafts, has been crucial.

Many others helped. Jim Forkel and the staff of the Princeton Tower Club, wherein I wrote every word of this thesis, made sure that the coffee machine was always working and that my workspace was comfortable. Bram Moolenaar and the other developers of `vim` made the best text editor in the world so that I could create and manage my content effectively. Those who worked on TeX and LaTeX made it possible for me to focus on creating content without worrying about formatting. Many have been willing to discuss these ideas with me and review my writing, and their attention has been most helpful. I owe special thanks to Mary Marshall for her careful reading of the first two chapters of this work and suggestions from a non–mathematical perspective. To my mother Britt–Marie and my father Jerome, who made me want to learn new things and then made it possible for me to do so, I owe all.

CONTENTS

LIST OF FIGURES

Introduction

When faced with a dynamic system, we often want to find an analytic, paper–and–pencil expression of how the system has evolved after time has elapsed. This is not always possible, however: some dynamic systems are governed by complicated rules that make analytic solutions impossible. In such a case, we might perform a numerical approximation by simulating the evolution of the system in a computer programmed with the governing laws. In a deterministic system—one that evolves the same way every time and leaves naught to chance—our computer will reveal the same approximation every time it performs the computation.

Non–deterministic systems—those whose evolution has some random influence—elude such precision. Every time that a stochastic system is realized, the outcome is different. No longer can we describe the way things look after a certain period of time. Instead, we can only offer a probability distribution of results, and, in particular, the expectation $\mathbb{E}[\cdot]$ of quantities in the system. In this work, we are interested in the way that a non–deterministic dynamic system shifts when we change its laws. In very general terms, we recognize some parameter λ that is used to generate a random variable X, and we try to find $\frac{\partial}{\partial\lambda}\mathbb{E}[X]$, the "$\lambda$-sensitivity" of X. In particular, we will be using the Malliavin calculus, the calculus of variations on stochastic processes, to make numerical approximation of $\frac{\partial}{\partial\lambda}\mathbb{E}[X]$ more efficient and accurate when an analytic solution is impossible.

We begin in chapter 1 by introducing Monte Carlo simulation in general and reviewing some classical approaches to computation of sensitivities like $\frac{\partial}{\partial\lambda}\mathbb{E}[X]$ with Monte Carlo methods. The chapter ends with a key observation in remark 1.2.5 of an "integration–by–parts" formula that looks like

$$\mathbb{E}\left[f'(X)\right] = \mathbb{E}\left[f(X)H\right] \tag{$\diamond$}$$

where H is called a "weight." This equation will motivate our use of the Malliavin calculus to find a similar but more effective formula. We take a detour in chapter 2 to introduce the tools and notation of Malliavin calculus from a theoretical standpoint, and we conclude that chapter by finding an integration–by–parts equation like ($\diamond$) that applies to a wide variety of problems. This formula is developed in proposition 2.3.1 In chapter 3, we apply this result to a problem in finance: the computation of sensitivities of the value of a derivative contract to parameters like stock price. These sensitivities are often known as "greeks." We conclude chapter 3 with numerical simulation to compare the effectiveness of our Malliavin estimators for the greeks with some of the classical methods we saw in chapter 1.

CHAPTER 1

Monte Carlo Methods

In this chapter, we discuss *Monte Carlo methods*, a set of tools to obtain numerical approximations of quantities in stochastic systems. Whenever we refer to Monte Carlo "methods" or "techniques" to obtain the expectation of a random variable X, we are referring to the procedure offered below:

```
FOR i = 1...N
   [Simulate a realization of X]
   X(i) = [Simulated value of X]
END
MonteCarloEstimate = SUM(X)/N
```

We will denote the output of this algorithm $\widehat{\mathbb{E}}[X]$, explicitly defined

$$\widehat{\mathbb{E}}[X] = \frac{1}{N}\sum_{i=1}^{N} X_i$$

where X_i is the value that X takes in the ith simulation. The strong law of large numbers (see [22, p. 9], for example) tells us that $\widehat{\mathbb{E}}[X] \to \mathbb{E}[X]$ as the number of simulations N grows large. Of course, large values of N do more than increase precision: they demand more computational power, more memory, and more time, facing us with a trade–off between the accuracy of our method and the resources it demands. This chapter will deal with this trade–off and the problems it creates.

When we perform Monte Carlo simulation, we seek the expected value of the random variable X. We are explicitly not interested in details of the distribution of X, its variance or higher moments. Thus, if we can find some random variable Y such that $\mathbb{E}[Y] = \mathbb{E}[X]$, our simulator could generate and average values for Y to construct $\widehat{\mathbb{E}}[Y]$, and the law of large numbers implies that $\widehat{\mathbb{E}}[Y] \to \mathbb{E}[X]$ for N large. In this case, we call Y an *unbiased estimator* for X. If we are able to find an unbiased estimator Y such that $\text{var}[Y] < \text{var}[X]$, then Y is preferable to X in our Monte Carlo simulation in the precise sense that we expect it to give more accurate results after a given number of simulations. This is because the variance of $\widehat{\mathbb{E}}[Y]$ will be smaller than the variance of $\widehat{\mathbb{E}}[X]$, i.e. the expected square of the difference between $\widehat{\mathbb{E}}[Y]$ and $\mathbb{E}[X]$ is smaller than the expected square of the difference between $\widehat{\mathbb{E}}[X]$ and $\mathbb{E}[X]$.

In an extreme example, suppose $X \sim \mathcal{N}(0,1)$ and $Y = 0$. The expectations of X and Y are identical. After N simulations

$$\mathrm{var}\left[\widehat{\mathbb{E}}\left[X\right]\right] = \mathrm{var}\left[\frac{1}{N}\left(X_1 + \ldots + X_N\right)\right] = \frac{1}{N}$$

with the assumption that the X_i are i.i.d.; on the other hand, $\mathrm{var}\left[Y\right] = 0$ trivially. Thus we prefer Y for Monte Carlo methods—a single approximation would yield the exact value we had sought without variance.

Of course, it may not be possible to find a good unbiased estimator Y in a given problem; then, we could seek a random variable Y whose expectation is near $\mathbb{E}\left[X\right]$. We would call Y a *biased estimator* and let $\mathrm{Bias}(Y) = \mathbb{E}\left[Y\right] - \mathbb{E}\left[X\right]$ in the formulation of [12, 22]. In a given application, a biased estimator with a smaller variance than the quantity being estimated could be useful if the bias were acceptably small.

We now present a question about computing sensitivities whose problematic solution will occupy this and subsequent chapters.

Question. *How can we use Monte Carlo methods to estimate the sensitivity of a random variable to a change in the laws that govern what value that quantity takes? That is, if we have a random variable X that depends on a real–valued parameter λ, how can we determine the change in $\mathbb{E}\left[X\right]$ that results from a perturbation of λ?*

A simple example can motivate the above question: suppose we were to blindly draw a colored ball from a bag and let $X = 1$ if it is red and $X = 0$ if it is blue . If the parameter $\lambda \in \mathbb{R}^+$ were the ratio of red to blue balls in the bag, changing λ would have an effect on $\mathbb{E}\left[X\right]$. We can see that $\mathbb{E}\left[X\right] = \lambda/(\lambda+1)$ in this case, and we can use the standard rules of the differential calculus to compute that

$$\frac{\partial}{\partial\lambda}\mathbb{E}\left[X\right] = \frac{(\lambda+1)-\lambda}{(\lambda+1)^2} = \frac{1}{(\lambda+1)^2}.$$

More generally, we will let $\psi(\lambda) = \mathbb{E}\left[X \mid \lambda\right]$ for some random variable X whose value depends on λ, and we will try to determine $\psi'(\lambda_0) = \frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda\right]$ when evaluated at $\lambda = \lambda_0$. For convenience, we adopt the notation $\psi'(\lambda_0) = \frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda_0\right]$.

1.1 Biased Estimators for Sensitivities

In this section, we use biased estimators to determine the sensitivity of a random variable to its parameters. That is—using the shorthand $\psi'(\lambda_0) = \frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda_0\right]$—we examine estimators Θ of $\psi'(\lambda_0)$ for which $\mathrm{Bias}(\Theta) = \mathbb{E}\left[\Theta - \psi'(\lambda_0)\right] \neq 0$. Bias is undesirable, surely, but the estimators in this section benefit from the fact that they are straightforward to compute and can be used to test the λ–sensitivity of virtually any random quantity.

Approach 1 (Forward Finite Difference)**.** A straightforward way to approximate the λ–sensitivity of a random quantity is the *finite difference* method. Given a function f that is differentiable at the point x_0, we know that its derivative $f'(x_0)$ is well–approximated by the slope of a secant line joining two points near x_0 on the curve $f(x)$; indeed, $h^{-1}(f(x_0+h) - f(x_0)) \to f'(x_0)$ as $h \to 0$.

Here, we will define the *forward finite difference estimator* $\hat{\Theta}_{F,h}$ (subscript "F" for "forward") to be

$$\hat{\Theta}_{F,h} = \frac{\widehat{\mathbb{E}}\left[X(\lambda_0+h)\right] - \widehat{\mathbb{E}}\left[X(\lambda_0)\right]}{h} = \frac{\widehat{\psi}(\lambda_0+h) - \widehat{\psi}(\lambda_0)}{h} \approx \psi'(\lambda_0)$$

for a small value of h.[1] As a Monte Carlo simulator performs an arbitrarily large number of simulations we have $\widehat{\mathbb{E}}\left[X\right] \to \mathbb{E}\left[X\right]$ almost surely, and we can similarly denote by $\Theta_{F,h}$ (without a caret) the value of $\hat{\Theta}_{F,h}$ as the number of simulations approaches infinity. Thus we have the theoretical estimator

$$\Theta_{F,h} = \frac{\mathbb{E}\left[X(\lambda_0+h)\right] - \mathbb{E}\left[X(\lambda_0)\right]}{h} = \frac{\psi(\lambda_0+h) - \psi(\lambda_0)}{h} \approx \psi'(\lambda_0),$$

which we could never compute in finite time as it requires infinite simulations. Understand that the final approximation here ("$\approx$") is *always* an approximation regardless of the number of simulations—even in the limit—because of the imprecision of using a finite difference to estimate the derivative. This is the source of bias, and we can quantify that bias by following [12, sec. 7.1]. Assuming ψ is twice differentiable, we can form the Taylor series of ψ near λ_0, giving us

$$\psi(\lambda_0+h) = \psi(\lambda_0) + \psi'(\lambda_0)h + \frac{1}{2}\psi''(\lambda_0)h^2 + o(h^2)$$

where $o(h^2)$ is "little–o" notation, i.e. it represents a term that goes to zero faster than some constant multiple of h^2. Rearranging terms,

$$\frac{\psi(\lambda_0+h) - \psi(\lambda_0)}{h} - \psi'(\lambda_0) = \frac{1}{2}\psi''(\lambda_0)h + o(h).$$

We recognize the estimator $\Theta_{F,h}$ as the ratio at left. Taking expectations,

$$\mathbb{E}\left[\Theta_{F,h} - \psi'(\lambda_0)\right] = \frac{\psi''(\lambda_0)h}{2} + o(h). \tag{1.1}$$

Equation (1.1) shows $\text{Bias}(\Theta_{F,h}) \neq 0$, i.e. that we have a biased estimator: for any nonzero h, our estimator will miss $\psi'(\lambda_0)$ by at least $\psi''(\lambda_0)h/2$ as our computers conduct arbitrarily many simulations. Fortunately, the bias decreases along with h to zero in the limit. Since we don't want an estimator to have bias, it seems we would like to set h very small to get the best estimator possible.

There is a drawback to a small h, however: as discussed above, $\Theta_{F,h}$ is a *theoretical* estimator that we approach with variance by computing $\hat{\Theta}_{F,h}$ with a finite number of simulations N. That is, $\hat{\Theta}_{F,h}$ is itself a random quantity with some variance around $\Theta_{F,h}$. We would prefer the variance to be small, of course, so that we can be more confident that our estimation $\hat{\Theta}_{F,h}$ is close to the theoretical estimator $\Theta_{F,h}$. Heuristically, Monte Carlo simulation will be better (less variance) the more simulations we perform, but the variance of $\hat{\Theta}_{F,h}$ also crucially depends on the value we choose for h [12]:

$$\text{var}\left[\hat{\Theta}_{F,h}\right] = \text{var}\left[\frac{\widehat{\psi}(\lambda_0+h) - \widehat{\psi}(\lambda_0)}{h}\right] = \frac{1}{h^2}\text{var}\left[\widehat{\psi}(\lambda_0+h) - \widehat{\psi}(\lambda_0)\right]. \tag{1.2}$$

[1] We re–emphasize here that the crowning carets indicate approximation via finite Monte Carlo methods.

The leading h^{-2} amplifies the variance of the estimator substantially for small h. The smaller our perturbation h, the less confident we can be that our Monte Carlo method is accurate. In heuristic terms, we are asking for large variance by dividing the very small, random quantity $\mathbb{E}[\psi(\lambda_0+h)] - \mathbb{E}[\psi(\lambda_0)]$ by the also small h.

As mentioned, this variance will decrease as the number of simulations N grows. To make this precise, we turn to (1.2) and follow the approach of [12] in analyzing $\operatorname{var}\left[\widehat{\psi}(\lambda_0+h) - \widehat{\psi}(\lambda_0)\right]$. Recall that a Monte Carlo procedure computes $\psi(\lambda) = \mathbb{E}[X \mid \lambda]$ by simulating $X_1, X_2, \ldots, X_N$ for large N and approximates $\mathbb{E}[X \mid \lambda] \approx \frac{1}{N}\sum_{i=1}^{N} X_i$. Thus

$$\begin{aligned}\operatorname{var}\left[\widehat{\psi}(\lambda_0+h) - \widehat{\psi}(\lambda_0)\right] &= \operatorname{var}\left[\frac{1}{N}\sum_{i=1}^{N} X_i(\lambda_0+h) - \frac{1}{N}\sum_{j=1}^{N} X_i(\lambda_0)\right] \\ &= \frac{1}{N^2}\operatorname{var}\left[\sum_{i=1}^{N}\left(X_i(\lambda_0+h) - X_i(\lambda_0)\right)\right].\end{aligned}$$

We now assume that the results of a pair of simulations are i.i.d., which is assuming that we are using truly random numbers to conduct our Monte Carlo simulation. With that assumption,

$$\frac{1}{N^2}\operatorname{var}\left[\sum_{i=1}^{N}(X_i(\lambda_0+h) - X_i(\lambda_0))\right] = \frac{1}{N^2}\left(N\operatorname{var}\left[X(\lambda_0+h) - X(\lambda_0)\right]\right)$$

which lets us rewrite (1.2) as

$$\operatorname{var}\left[\hat{\Theta}_{F,h}\right] = \frac{1}{Nh^2}\operatorname{var}\left[X(\lambda_0+h) - X(\lambda_0)\right]. \tag{1.3}$$

We interpret (1.3) in terms of the expression for the bias of the $\Theta_{F,h}$ in (1.1). We saw there that bias is linearly decreasing as $h \to 0$; here we see that variance is quadratically increasing as $h \to 0$. As a result, to get a less–biased estimate and maintain a similar confidence in our estimate, we need to increase the number of simulations N as the square of our decrease in h. There is a discussion in [12, p. 381–384] of how one could optimize the trade–off between bias and variance (i.e. the problem of choosing h) using mean square error objective functions.

We have now rigorously established the forward finite difference estimator and determined its bias and variance characteristics. Note that our technique required no knowledge of the distribution of X or any other quantity—all we need is the ability to simulate the variable X a large number of times and a powerful computer to do so.

Approach 2 (Other Finite Difference Methods)**.** In the previous approach, $\Theta_{F,h} = \frac{\psi(\lambda_0+h)-\psi(\lambda_0)}{h}$ was the forward difference estimator in that looked "forward" to λ_0+h to make a linear approximation. The *backward difference estimator* is similarly conceived , with $\Theta_{B,h} = \frac{\psi(\lambda_0)-\psi(\lambda_0-h)}{h}$. This approach has the same bias and variance characteristics as its forward relative.

Another approach uses the *central difference estimator*—which is mathematically equivalent to the average of the forward and backward difference estimators—defined by $\Theta_{C,h} = \frac{\psi(\lambda_0+h)-\psi(\lambda_0-h)}{2h}$. As Glasserman [12, p. 79] points out, this is an improvement over Θ_F and Θ_B as it has a bias on the smaller order of h^2. This fact is proved by writing out the Taylor series for the estimator, in which the forward and backward linear terms cancel.

This method seems to come with no additional computational cost, as we are still only taking a difference between two simulated points on the curve $\psi(\lambda)$. As a practical matter, however, any application that demanded the computation of $\psi'(\lambda_0)$ would probably be interested in the value of the expectation itself (i.e. would want to know $\psi(\lambda_0)$ in addition). Computing the expectation *and* its sensitivity to changes in λ requires three simulations with the central finite difference method, a 50% increase in overhead.

Approach 3 (High–Order Polynomial Approximation)**.** The finite difference methods assume that the function $\psi(\lambda)$ is well–approximated by a linear function in a small neighborhood of λ_0, and they compute and differentiate that linear function to estimate $\psi'(\lambda_0)$. We can improve by using a $n > 1$ degree polynomial $q(x)$ instead of a straight line (which is of degree $n = 1$) to approximate $\psi(\lambda)$ in the region of interest; we could then differentiate q at λ_0 to obtain the estimator Θ_{HOPA} for $\psi'(\lambda_0)$. In theory, a higher degree polynomial provides a better representation of ψ and thus yields a less biased approximation of ψ'. We will specify that we want to use an seventh degree polynomial for our approximation of ψ. Choosing $n = 7$ for the degree of the high–order polynomial is of course arbitrary; we will shortly see that the choice of n gives us a trade–off between computability and bias.

To find an n degree polynomial q that fits the function ψ, we need to sample ψ in at least $n + 2$ places and minimize $\|q - \psi\|$ locally in the L^2 sense. We need $n + 2$ points to exhaust the $n + 1$ degrees of freedom that an n degree polynomial has in order to get a useful interpolating function.[2] We will be using Monte Carlo methods to approximate $\psi(\lambda)$ for nine values $\lambda = \lambda_0 + r$ where $r \in [-\epsilon, \epsilon]$.[3] We can specify to choose the nine values for r evenly distributed in $[-\epsilon, \epsilon]$ as a matter of convenience. Example 1.1.1 provides an example of this sort of computation. All details aside, it is evident that we will need to perform nine Monte Carlo simulations to compute our approximation of $\psi'(\lambda_0)$, which is substantial computational overhead.

To compute the bias of this approach, we suppose that the function $\psi(\lambda)$ were smooth near λ_0 and compute its Taylor series to n terms at λ_0. The resulting n degree polynomial will be the best n degree polynomial to approximate ψ in an ϵ–neighborhood of λ_0: its error will be $o(\epsilon^{n+1})$, and any other n degree polynomial will differ from ψ on at some term of order $k \leq n$, producing an error of at least $o(\epsilon^k)$. It follows that the Θ_{HOPA} estimator with $n = 7$ will have a bias of $o(\epsilon^8)$ where ϵ is (as above) the radius around λ_0 in which we choose our sample points for Monte Carlo simulation.

The high–order polynomial approach, has advantages over the finite difference method in that it greatly reduces the bias by not using a linear model for the derivative; an obvious disadvantage is the computational power and time needed to perform Monte Carlo simulation nine times.

Example 1.1.1. [Illustrating approach 3] Suppose we have two random variables, X and Y, each uniformly distributed on $[0, 1]$. Let $V = \cos\left(\mathbf{1}_{X>0.5} e^{\lambda Y}\right)$. Let $\psi(\lambda) = \mathbb{E}[V]$ given that V is computed with the specified value of λ. We perform Monte Carlo simulation to obtain values for V for nine different values of λ near $\lambda = \lambda_0 = 0.1$ (the `MATLAB` code for this computation can be found in appendix B). We then used `MATLAB`'s curve–fitting toolbox to find a seventh order polynomial to fit the data. Figure 1.1 shows the Monte Carlo results and the approximating polynomial.

[2] The mechanics of fitting a polynomial to data and defining what makes a "good" fit are beyond the scope of this work. See [15], for example, for a generalized framework.

[3] As in the finite difference method, smaller values for ϵ produce less bias—as we shall see in the next paragraph—but larger variance for the estimator as they magnify the inherent imprecision of Monte Carlo methods

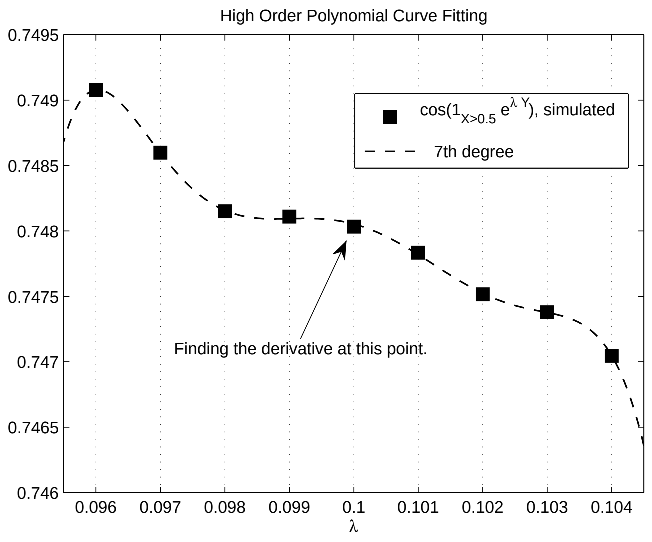


Figure 1.1: Fitting a high–order polynomial to Monte Carlo–simulated data for the purpose of finding a sensitivity with respect to the parameter λ, as in example 1.1.1.

Approximating the derivative at $\lambda_0 = 0.1$ is accomplished by differentiating the polynomial shown in that figure at λ_0.

Remark 1.1.2. Some applications require the computation of second derivatives such as $\psi''(\lambda) = \frac{\partial^2}{\partial\lambda^2}\mathbb{E}[X]$ or cross derivatives such as $\frac{\partial^2}{\partial\lambda_1\partial\lambda_2}\mathbb{E}[X]$. The difference methods we have discussed in this section could be adapted to compute these second order sensitivities by computing first order sensitivities at two different points using the difference methods above and applying a finite difference method a second time on those approximated sensitivities. This leads to an equation like

$$\psi''(\lambda_0) \approx h^{-1}\left(\frac{\psi(\lambda_0+h)-\psi(\lambda_0)}{h} - \frac{\psi(\lambda_0)-\psi(\lambda_0-h)}{h}\right) = \frac{\psi(\lambda_0+h)-2\psi(\lambda_0)+\psi(\lambda_0-h)}{h^2}.$$

Glasserman points out that the difference approach is fundamentally bad at computing these second order sensitivities [12]: the variance of the above estimator is on the order of h^{-4}, which can be extremely large for h that are small enough to give useful estimates.

The higher–order polynomial approximation is more helpful. If we choose the degree of the approximating polynomial q to be large enough, we can differentiate q any number of times and keep the variance and bias under control. Again, this comes at the cost of computational resources.

The finite difference and polynomial approximation estimators are biased because they are inherently imprecise ways to calculate derivatives. We have discussed the trade–off between computational time (the variance of the estimator) and bias, and, while techniques exist to reduce the sting of this trade–off, these methods will always be imprecise. That said, these approaches always work in the sense that given enough time and computational power, reasonably accurate results can be had regardless of the complexity of the model. As long as we know enough about the random variable X to simulate it, we can do so with different values a parameter λ and use finite differences or higher–order models to approximate the λ-sensitivity of X.

1.2 Unbiased Estimators for Sensitivities

In this section, we discuss *unbiased* estimators Θ for the sensitivity of $\mathbb{E}[X]$ to the parameter λ. With such Θ, we no longer face a trade–off between bias and computational time. This does not mean that we can necessarily perform fewer simulations—doing so should always improve Monte Carlo methods by reducing estimator variance—but we can be almost sure that the value we approach is accurate.

We will focus on two specific approaches to unbiased Monte Carlo estimation of sensitivities, the "pathwise derivative" and "likelihood" methods. Though each is limited by the narrower scope of problems it can successfully address, each is powerful in the cases that it can be applied. Elements of each of these methods will motivate our search for a powerful unbiased estimator using the Malliavin calculus.

Approach 4 (Pathwise Derivative Estimates)**.** Liebnitz's rule for differential calculus establishes the useful technique of exchanging the order of integration and differentiation. That is, given certain conditions of continuity and convergence of the integral in question, we have $\int \partial f = \partial \left(\int f \right)$. In the context of probability theory, when a random variable (a function from a probability space Ω to $\mathbb{R}$) has finite expectation, we can often express the derivative of its expectation as the expectation of its derivative. In particular,

$$\frac{\partial}{\partial \lambda}\mathbb{E}[X \mid \lambda_0] = \frac{\partial}{\partial \lambda}\int_{-\infty}^{\infty} x(\lambda)p_X(x)dx\bigg|_{\lambda=\lambda_0} = \int_{-\infty}^{\infty}\left(\frac{\partial}{\partial \lambda}x(\lambda)\right)\bigg|_{\lambda=\lambda_0} p_X(x)dx = \mathbb{E}\left[\frac{\partial}{\partial \lambda}X \;\middle|\; \lambda_0\right]$$

when the Liebnitz rule applies.[4] In general, we will assume that the Liebnitz rule applies, i.e. that $\mathbb{E}[X \mid \lambda]$ is continuous in λ and that the expectations in question are bounded. To clarify the above equation, $\frac{\partial}{\partial \lambda}X$ indicates the derivative of X with respect to λ under a fixed realization of X. That is to say that we fix the chance parameter ω and isolate the effect of changing λ, which is the natural way to approach the problem [12].

The chain rule of calculus generalizes the above equation to functions of random variables: if $f : \mathbb{R} \to \mathbb{R}$ is a real–valued function, then we can write

$$\frac{\partial}{\partial \lambda}\mathbb{E}[f(X) \mid \lambda_0] = \mathbb{E}\left[\frac{\partial}{\partial \lambda}f(X) \;\middle|\; \lambda_0\right] = \mathbb{E}\left[f'(X)\frac{\partial}{\partial \lambda}X \;\middle|\; \lambda_0\right] \tag{1.4}$$

again assuming Liebnitz's rule holds. The transformation in (1.4) will be the crucial one for the pathwise derivative method. In particular, (1.4) shows that we have

$$\Theta_{PD} = f'(X)\frac{\partial}{\partial \lambda}X,$$

an unbiased estimator for $\frac{\partial}{\partial \lambda}\mathbb{E}[f(X) \mid \lambda_0]$. We have thus written our problem in a form that a Monte Carlo simulator could handle directly. See [25], [12], and [14] for rigorous constructions of pathwise derivative methods.

This pathwise derivative method captures our desire in the previous section to send $h \to 0$ to obtain the true derivative. In that section we could not let $h \to 0$ because of the implications for

[4] An important note on this formulation: we are explicitly constructing the density function of X not to depend on the non–stochastic parameter λ. For example, if $X \sim \mathcal{N}(\lambda, 1)$, we could carefully write $X = Z + \lambda$ where $Z \sim \mathcal{N}(0,1)$ and then say $\frac{\partial}{\partial \lambda}\mathbb{E}[X(\lambda)] = \int_{\mathbb{R}}\left(\frac{\partial}{\partial \lambda}(z+\lambda)\right)N(z)dz$ (note that this evaluates to $\int_{\mathbb{R}} N(z)dz = 1$, the correct answer).

the variance of our estimator, but here we are not so affected because we use the derivative of a deterministic function instead of a finite difference of two approximations of random functions. Of course, we need the derivative $\frac{\partial}{\partial\lambda}X$ to exist almost everywhere, but this requirement is satisfied in most applications. Indeed, this method is very useful in the solutions of many different problems; see [12, p. 386–401] for a discussion of some of them. The technique, however, is not without limitations. The next proposition and its generalization demonstrate a class of problems for which the pathwise derivative technique fails to provide meaningful results.

Proposition 1.2.1. *Let X be a random variable that takes values in $\mathbb{R}$ and depends on a parameter λ. Let $H(x) = \mathbf{1}_{x>0}$. Then the pathwise derivative technique of Monte Carlo simulation will fail to be useful in that it will estimate $\frac{\partial}{\partial\lambda}\mathbb{E}\left[H(X)\right] = 0$.*

Proof. Formally, $\frac{\partial}{\partial\lambda}H$ is nonzero on a set that is so small that a Monte Carlo simulator almost always misses it, almost surely forcing our estimate to be zero.

More rigorously, we follow the pathwise derivative technique and apply the Liebnitz rule and the chain rule of calculus to obtain

$$\frac{\partial}{\partial\lambda}\mathbb{E}\left[H(X)\right] = \mathbb{E}\left[\frac{\partial}{\partial\lambda}H(X)\right] = \mathbb{E}\left[H'(X)\frac{\partial}{\partial\lambda}X\right] = \mathbb{E}\left[\delta_0(X)\frac{\partial}{\partial\lambda}X\right]$$

where $\delta_0(x)$ is the Dirac δ function, i.e. the function specified by $\lim_{\epsilon\to 0}\delta_0^\epsilon(x) = \lim_{\epsilon\to 0}\frac{1}{2\epsilon}\mathbf{1}_{x<|\epsilon|}$.[5] We can verify that $H'(x) = \delta_0(x)$ by computing

$$\int_{-\infty}^{x}\delta_0(t)\ \mathrm{d}t = \underbrace{\int_{-\infty}^{x}\lim_{\epsilon\to 0}\delta_0^\epsilon(t)\ \mathrm{d}t = \lim_{\epsilon\to 0}\int_{-\infty}^{x}\delta_0^\epsilon(t)\ \mathrm{d}t}_{\text{justified by Fatou–Lebesgue, see [23].}} = \lim_{\epsilon\to 0}\int_{-\infty}^{x}\frac{1}{2\epsilon}\mathbf{1}_{t<|\epsilon|}\ \mathrm{d}t$$

$$= \begin{cases}\lim_{\epsilon\to 0} 0 & x < -\epsilon\\ \lim_{\epsilon\to 0}\frac{x+\epsilon}{2\epsilon} & |x| < \epsilon\\ \lim_{\epsilon\to 0} 1 & x > \epsilon\end{cases}$$

$$\to H(x)$$

We thus have $H'(X) = \delta_0(X) = 0$ almost surely, which means that in the expression

$$\frac{\partial}{\partial\lambda}\mathbb{E}\left[H(X)\right] = \mathbb{E}\left[\delta_0(X)\frac{\partial}{\partial\lambda}X\right]$$

the inside of the expectation operator on the right–hand side is zero unless $X = 0$ exactly. This is crucially *not* to say that the sensitivity $\frac{\partial}{\partial\lambda}\mathbb{E}\left[H(X)\right]$ is zero, but it does have serious consequences for Monte Carlo simulation.

If we use an ideal random number generator to simulate values of X, we find that $X \neq 0$ almost surely. This is because X is supported on a set of positive measure, which makes $\mathbb{P}\left\{X = r\right\} = 0$ for any real value r. When we perform Monte Carlo simulation of X for some finite number of trials, we expect to never see $X = 0$ among our results. Thus all of our simulations of $\frac{\partial}{\partial\lambda}H(X)$ will equal zero with probability one, and our Monte Carlo estimate of $\frac{\partial}{\partial\lambda}\mathbb{E}\left[H(X)\right]$ will accordingly be zero. $\square$

[5] Note that $\int_{\mathbb{R}}\delta_0^\epsilon(x)\ \mathrm{d}x = 1$ for all ϵ.

This mechanism of the above proof easily generalizes to hold for any function with jump discontinuities, giving us the following proposition.

Proposition 1.2.2 (Limitations of Pathwise Derivative Method)**.** *Let X be a random variable that takes values in $\mathbb{R}$ and depends on a parameter λ. Let $f(x)$ be a function of x with at least one jump discontinuity, i.e. there exists some x_0 such that*

$$\lim_{x \to x_0^-} f(x) \neq \lim_{x \to x_0^+} f(x).$$

Then the pathwise derivative technique of Monte Carlo simulation will not accurately compute $\frac{\partial}{\partial \lambda}\mathbb{E}\left[f(X)\right]$.

The issues raised by these two propositions are important because the pathwise derivative method is incredibly useful for computing sensitivities when it is applicable. In chapter 3, section 3.4, we will revisit this issue.

Remark 1.2.3. Practically, one might try to avoid the issue raised in proposition 1.2.2 by smoothing a jump discontinuity in f at x_0 with a continuous function $\tilde{f}$ that differs from f only near x_0 using a version of Littlewood's principles [23]. Then $\tilde{f}'$ will not have a singularity, and Monte Carlo simulation will no longer miss the jump. This is far from a perfect solution, however. For one thing, our estimator is no longer unbiased as $\tilde{f}$ differs from f. For another, if we make $\tilde{f}$ very steep in the portion where it approximates f, Monte Carlo simulation may still fail if it misses the narrow window where $\tilde{f}'$ is positive and large.

The issue brought to light in proposition 1.2.2 is a general criticism of the inefficiency of Monte Carlo methods. If a function of a random variable spikes and returns quickly on a sufficiently small domain (the Dirac δ function is an extreme example), a Monte Carlo simulator can miss this spike and produce approximations that do not approach the desired result. The next approach alleviates this problem by finding an unbiased estimator to use in simulation that "smooths out" any offending spikes.

Approach 5 (The Likelihood Method)**.** In our derivation of the pathwise derivative approach, we had a random variable X that depended on λ. However, we carefully noted—see footnote 4 of this chapter—that X was computed as some deterministic transform of a random quantity and that λ affected that transform, not the distribution of that underlying random quantity. The *likelihood method*, which we shall discuss now, takes the opposite approach by, as Broadie and Glasserman [5, sec. 2.2] call it, putting "the dependence on the parameter of interest in an underlying probability density rather than in a random variable."

To make this distinction clear, we return to the example in footnote 4: suppose $X \sim \mathcal{N}(\lambda, 1)$. Then

- when using the pathwise derivative method, we would write $X = Z + \lambda$ where $Z \sim \mathcal{N}(0, 1)$. The underlying random quantity would be Z where $p_Z(z) = (\sqrt{2\pi})^{-1}\exp(-z^2/2)$, and we note that this distribution does not depend on λ; whereas,

- when using the likelihood method, X is itself the underlying random quantity and $p_{X,\lambda}(x) = (\sqrt{2\pi})^{-1}\exp(-(z-\lambda)^2/2)$, which does depend on λ.

We now develop the likelihood method for unbiased Monte Carlo approximation of the sensitivity of the expectation of a random variable. The development here is a more general exposition of the argument in [5]; the reader should be aware that other formulations exist with the same name. Suppose we have some random variable X with known distribution $p_{X,\lambda}(x)$ that depends on a parameter λ and some function $f : \mathbb{R} \to \mathbb{R}$. We write

$$\mathbb{E}\left[f(X) \mid \lambda_0\right] = \int_{\mathbb{R}} f(x) p_{X,\lambda}(x) \, \mathrm{d}x \bigg|_{\lambda_0}$$
$$\frac{\partial}{\partial \lambda}\mathbb{E}\left[f(X) \mid \lambda_0\right] = \int_{\mathbb{R}} \frac{\partial}{\partial \lambda}\left(f(x) p_{X,\lambda}(x)\right)\bigg|_{\lambda_0} \mathrm{d}x = \int_{\mathbb{R}} f(x) \, \frac{\partial}{\partial \lambda} p_{X,\lambda}(x)\bigg|_{\lambda_0} \mathrm{d}x$$

where the last equality is justified by the fact that f does not depend on λ. We now use the chain rule to assert that

$$\frac{\partial}{\partial \lambda} \log p_{X,\lambda}(x) = \frac{1}{p_{X,\lambda}(x)} \frac{\partial}{\partial \lambda} p_{X,\lambda}(x)$$
$$\left(\frac{\partial}{\partial \lambda} \log p_{X,\lambda}(x)\right) p_{X,\lambda}(x) = \frac{\partial}{\partial \lambda} p_{X,\lambda}(x),$$

and we insert this identity to write the key transformation of the likelihood method:

$$\frac{\partial}{\partial \lambda}\mathbb{E}\left[f(X) \mid \lambda_0\right] = \int_{\mathbb{R}} \left(f(x) \frac{\partial}{\partial \lambda} \log p_{X,\lambda}(x)\right) p_{X,\lambda}(x)\bigg|_{\lambda_0} \mathrm{d}x. \tag{1.5}$$

Remark 1.2.4. Before we discuss why (1.5) is helpful, a note on its validity: for that equation to hold, we must have $p_{X,\lambda}(x)$ strictly positive on the domain of integration so that $\log p_{X,\lambda}(x)$ is real–valued. This not a major restriction on the usefulness of our method because the subset of $\mathbb{R}$ where $p_{X,\lambda}(x) = 0$ can be safely omitted from the integral's domain when computing expectations. That is, we can write $\mathbb{E}\left[f(X) \mid \lambda_0\right] = \int_{A(X)} f(x) p_{X,\lambda}(x) \, \mathrm{d}x$ where $A(X) = \{r \in \mathbb{R} \, : \, p_{X,\lambda}(x) > 0\}$. In turn, (1.5) could be rewritten

$$\frac{\partial}{\partial \lambda}\mathbb{E}\left[f(X) \mid \lambda_0\right] = \int_{A(X)} \left(f(x) \frac{\partial}{\partial \lambda} \log p_{X,\lambda}(x)\right) p_{X,\lambda}(x)\bigg|_{\lambda_0} \mathrm{d}x$$

if $p_{X,\lambda}$ were zero on some subset of $\mathbb{R}$.

Another note on (1.5) is that the density $p_{X,\lambda}$ has to be differentiable with respect to λ on the relevant domain. In practice, this requirement will usually be satisfied.

Returning to our exposition, the transformation (1.5) is incredibly useful as its right hand side is an expectation of a function of the random variable X. We can rewrite (1.5) as

$$\frac{\partial}{\partial \lambda}\mathbb{E}\left[f(X) \mid \lambda_0\right] = \mathbb{E}\left[f(X) \frac{\partial}{\partial \lambda} \log p_{X,\lambda} \,\middle|\, \lambda_0\right].$$

This final statement gives us an unbiased estimator Θ_L for the sensitivity of $\mathbb{E}\left[f(X)\right]$ to the parameter λ given by

$$\Theta_L = f(X) \frac{\partial}{\partial \lambda} \log p_{X,\lambda}.$$

To implement a Monte Carlo approximation, we first simulate X a number of times with the parameter λ_0 and find $f(X)$ in each case. We also compute $\log p_{X,\lambda}(X)$, and take the appropriate product to determine $\frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda_0\right]$. For some examples of how this can be used in real problems, see [12, p. 401–418].

One of the major limitations of this method is that it requires us to know the density $p_{X,\lambda}$ explicitly as a function of λ and to be able to differentiate it with respect to λ. We will not, as a general rule, know these things for complicated problems, which is a serious limitation. In such a situation, the likelihood method fails to help.

Remark 1.2.5 (Integration–by–Parts)**.** In this section, we have developed two unbiased estimators for $\frac{\partial}{\partial\lambda}\mathbb{E}\left[f(X) \mid \lambda_0\right]$. Each has its limitations, as we have discussed at length, but the law of large numbers implies that when we perform a large number of simulations, each estimator will converge on the correct value. In particular, we have the estimators

$$\Theta_{PD} = \frac{\partial}{\partial\lambda} f'(X)$$

and

$$\Theta_L = f(X)\frac{\partial}{\partial\lambda}\log p_{X,\lambda}(X),$$

as we have seen above. Since these estimators are both unbiased with regard to the same quantity, they have the same expected value, i.e.

$$\mathbb{E}\left[\frac{\partial}{\partial\lambda}f(X) \;\middle|\; \lambda_0\right] = \mathbb{E}\left[f(X)\frac{\partial}{\partial\lambda}\log p_{X,\lambda} \;\middle|\; \lambda_0\right]. \tag{1.6}$$

We will call this type of equation a stochastic integration–by–parts formula, a way to change the expectation of the derivative of some function into the expectation of the function itself times a "weight." For now, that weight is a function of the density of X. Once we have developed the fundamentals of the Malliavin calculus in the next chapter, however, we will be able to develop a series of formulas that let us compute "Malliavin weights" that allow us to write an equation like (1.6) without knowing $p_{X,\lambda}$.

CHAPTER 2

The Malliavin Calculus

The Malliavin calculus, developed by Paul Malliavin in 1976 [17], is the calculus of variations for stochastic processes. A stochastic process is a collection of random variables that are thought of as being indexed by time; such a process could be denoted $\{F_t\}_{t\in[0,T]}$ (see [22], for example, for a general introduction). For a specified t_0, F_{t_0} maps a probability space Ω to $\mathbb{R}$, and one would hope to be able to understand how small changes in the chance parameter $\omega \in \Omega$ affect the value that F_{t_0}. The Malliavin calculus makes this question precise, tells us under which circumstances we can answer it, and gives us the tools necessary to do so.

Recalling the discussion in chapter 1, the likelihood method gave us a way to compute the sensitivity of a random quantity X to some parameter λ. The approach was effective because it explicitly made the probability distribution of X dependent on λ and computed the partial derivative of p_X with respect to λ. The drawback to the method was the requirement that we had to know p_X explicitly to understand how changing λ affected it. The Malliavin calculus tools that we develop in this chapter will allow us to invent techniques similar to the likelihood method that we can use to perform efficient Monte Carlo simulation of $\frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda_0\right]$ even when we do not know p_X. Malliavin calculus was not developed to solve this type of problem, but authors like Fournié et. al. [9] began to realize the potential use—in particular for problems in mathematical finance—in the late 1990s after the 1995 publication of Nualart's canonical text on the subject ([20]).

The tools that we will introduce in this chapter are the Malliavin derivative operator D and its adjoint $\delta(\cdot)$, the latter of which coincides with a generalization of the Itô stochastic integral. We will be working with a special class of random variables that can be expressed as a smooth function of a Brownian motion. Specifically, suppose $W = \{W_t\}_{t\in[0,T]}$ is a Brownian motion on some probability space $(\Omega, \mathcal{F}, P)$, and assume $\mathcal{F}$ is generated by W. We define $\mathcal{S}$ as the set of random variables F such that

$$F = f(W_{t_1}, \ldots, W_{t_n})$$

for some some sufficiently smooth (C^∞) real function $f : \mathbb{R}^n \to \mathbb{R}$.

2.1 The Malliavin Derivative

The Malliavin derivative operator D defines the derivative of a random variable with respect to the "chance parameter" $\omega \in \Omega$, i.e. the change in that random variable as a result of a small perturbation in ω. Given a random variable $F \in \mathcal{S}$ as defined above, we formally think of a small perturbation in ω as a small change in the Brownian motion W that F is derived from. In a heuristic sense, computing DF is like finding "$\frac{\partial F}{\partial W}$" [16].

As W is a process, we can perturb it at many points; accordingly, we can measure F's sensitivity to changes in W_t for any t in our time domain. For this reason, the Malliavin derivative DF is a process—as opposed to a single value—that measures the sensitivity of F to small changes in W_t for a range of values of t. We write $DF = \{D_t F\}_{t\in[0,T]}$, and we define the operator D_t on the random variable $F = f(W_{t_1}, \dots, W_{t_n})$ as

$$D_t F = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} f(W_{t_1}, \dots, x_i, \dots, W_{t_n})\bigg|_{x_i = W_{t_i}} \mathbf{1}_{t<t_i}. \tag{2.1}$$

This definition merits short discussion. The function f takes as its arguments some number $n \geq 1$ of values from the diffusion W, and we compute the partial derivative of f with respect to each of its arguments. Before summing those partial derivatives, we apply the indicator function $\mathbf{1}_{t<t_i}$ to the ith term where t_i is the time in the Wiener process that f uses for its ith argument. That is, if our random variable were $F = f(W_{T/2}, W_T)$, the Malliavin derivative would be

$$D_t F = \left(\frac{\partial}{\partial x_1} f(x_1, W_T)\mathbf{1}_{t<T/2} + \frac{\partial}{\partial x_2} f(W_{T/2}, x_2)\mathbf{1}_{t<T} \right)\bigg|_{x_1 = W_{T/2}, x_2 = W_T}$$

As promised, D_t is a calculation of the extent to which F changes when we change W_t but leave $W_T - W_t$ the same.

It is worth noting that there are other definitions of D that can be shown to be equivalent to the above, but the one we use—found in [9] and in a slightly more general form in [20]—is the most convenient for the computations we will be performing.

Example 2.1.1. The simplest example of the operation of D is the computation of the Malliavin derivative of the random variable $F = W_{t_0}$ for some fixed value of t_0. It is trivial to check that $F \in \mathcal{S}$ in this case as

$$F = f(W_{t_0})$$

where f is the identity function (which meets the necessary smoothness conditions). Thus $n = 1$ in (2.1). Then

$$D_s W_{t_0} = \frac{\partial}{\partial x} x\bigg|_{x = W_{t_0}} \mathbf{1}_{s<t_0} = \mathbf{1}_{s<t_0} \tag{2.2}$$

So the derivative process is

$$\{D_s F\} = \begin{cases} 1 & s < t_0 \\ 0 & s \geq t_0, \end{cases}$$

which is to say that $F = W_{t_0}$ is sensitive (indeed, with direct proportionality) to changes in W_s when $s < t_0$ but unaffected by changes in W_s when $s > t_0$.

This result makes perfect sense in terms of how a Wiener process is defined. If $s < t_0$, then $\mathbb{E}\left[W_{t_0} - W_s \mid W_s\right] = 0$ (it is a martingale) and we expect a change in W_s to yield an equivalent change in W_t. If $s > t_0$, no such dependence exists, so it makes sense that the derivative is zero in that region.

We now prove several properties of D that will be useful for the computations we perform. These properties will help to establish an integration–by–parts rule in the final section of this chapter, and they will make the use of that rule easier in practice. The next proposition rigorously demonstrates properties that D inherits from its definition in terms of partial derivatives.

Proposition 2.1.2. *The D operator is*

- *linear, i.e.*
$$D(aF + bG) = aDF + bDG,$$
- *respects the chain rule, i.e.*
$$D(\phi(F)) = \phi'(F)DF$$
where ϕ is a real valued function and ϕ' represents its derivative in the typical sense, and
- *respects the product rule, i.e.*
$$D(FG) = FDG + GDF.$$

Proof. First, we show linearity. Let $F = aX + bY$ where $a, b \in \mathbb{R}$. The random variables X, Y are members of $\mathcal{S}$ (i.e. they are functions of Wiener processes), and in particular, we have $X = f_1(W_{r_1}, \ldots, W_{r_{m_1}})$ and $Y = f_2(W_{s_1}, \ldots, W_{s_{m_2}})$. To simplify notation, we collect all the times $r_1, \ldots, r_{m_1}, s_1, \ldots, s_{m_2}$, relabel them $t_1, \ldots, t_N$, and write

$$\begin{aligned} X &= \tilde{f}_1(W_{t_1}, \ldots, W_{t_N}) \\ Y &= \tilde{f}_2(W_{t_1}, \ldots, W_{t_N}) \end{aligned}$$

noting that X (or Y) may not depend on some of the t_j. Then

$$\begin{aligned} D_tF &= D_t(aX + bY) \\ &= \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left[a\tilde{f}_1(W_{t_1}, \ldots, x_i, \ldots, W_{t_N}) + b\tilde{f}_2(W_{t_1}, \ldots, x_i, \ldots, W_{t_N})\right]\Big|_{x_i = W_{t_i}} \mathbf{1}_{t<t_i} \\ &= \left(a \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \tilde{f}_1(W_{t_1}, \ldots, x_i, \ldots, W_{t_N})\Big|_{x_i = W_{t_i}} \mathbf{1}_{t<t_i}\right) \\ &\quad + \left(b \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \tilde{f}_2(W_{t_1}, \ldots, x_i, \ldots, W_{t_N})\Big|_{x_i = W_{t_i}} \mathbf{1}_{t<t_i}\right) \end{aligned}$$

where we have crucially used the linearity of the partial derivative operator. Now, for the t_i that do not equal s_j for any j, we have $\frac{\partial}{\partial x_i}\tilde{f}_1(W_{t_1}, \ldots, x_i, \ldots, W_{t_N}) = 0$ because X is independent of

W_{t_i}. A similar statement is true for the t_i upon which Y does not depend. We can thus rewrite our expression for D_tF as

$$\begin{aligned}D_tF &= \left(a \sum_{i=1}^{m_1} \frac{\partial}{\partial x_i}\ f_1(W_{r_1},\dots,x_i,\dots,W_{r_{m_1}})\big|_{x_i=W_{r_i}}\ \mathbf{1}_{t<t_i} \right) \\ &\quad + \left(b \sum_{i=1}^{m_2} \frac{\partial}{\partial x_i}\ f_2(W_{s_1},\dots,x_i,\dots,W_{s_{m_2}})\big|_{x_i=W_{s_i}}\ \mathbf{1}_{t<t_i} \right) \\ &= aD_tX + bD_tY.\end{aligned}$$

This concludes the proof of linearity.

As for the chain rule, suppose ϕ is a real valued function and $F \in \mathcal{S}$ with $F = f(W_{t_1},\dots,W_{t_n})$. Let $G = \phi(F)$. Then

$$\begin{aligned}D_tG &= \sum_{i=1}^{n} \frac{\partial}{\partial x_i}\phi(f(W_{t_1},\dots,x_i,\dots,W_{t_n}))\bigg|_{x_i=W_{t_i}} \mathbf{1}_{t<t_i} \\ &= \sum_{i=1}^{n} \phi'(f(W_{t_1},\dots,W_{t_n}))\frac{\partial}{\partial x_i} f(W_{t_1},\dots,W_{t_n})\bigg|_{x_i=W_{t_i}} \mathbf{1}_{t<t_i} \\ &= \phi'(f(W_{t_1},\dots,W_{t_n}))\sum_{i=1}^{n} \frac{\partial}{\partial x_i} f(W_{t_1},\dots,x_i\dots,W_{t_n})\bigg|_{x_i=W_{t_i}} \mathbf{1}_{t<t_i} \\ &= \phi'(F)D_tF,\end{aligned}$$

concluding the proof of the chain rule for the operator D.

We finally demonstrate the product rule. Suppose we have random variables X, Y in $\mathcal{S}$ with $X = f_1(W_{r_1},\dots,W_{r_{m_1}})$ and $Y = f_2(W_{s_1},\dots,W_{s_{m_2}})$. As we did for the proof of linearity, we simplify notation by collecting all the times $r_1,\dots,r_{m_1}, s_1,\dots,s_{m_2}$, relabeling them $t_1,\dots,t_N$, and writing

$$\begin{aligned}X &= \tilde{f}_1(W_{t_1},\dots,W_{t_N}) \\ Y &= \tilde{f}_2(W_{t_1},\dots,W_{t_N}).\end{aligned}$$

Then

$$\begin{aligned}D_t(XY) &= \sum_{i=1}^{n} \frac{\partial}{\partial x_i}\tilde{f}_1(W_{t_1},\dots,x_i,\dots,W_{t_n})\tilde{f}_2(W_{t_1},\dots,x_i,\dots,W_{t_n})\bigg|_{x_i=W_{t_i}} \mathbf{1}_{t<t_i} \\ &= \sum_{i=1}^{n} \left(\tilde{f}_1(W_{t_1},\dots,W_{t_n})\frac{\partial}{\partial x_i}\tilde{f}_2(W_{t_1},\dots,x_i,\dots,W_{t_n})\bigg|_{x_i=W_{t_i}} \right. \\ &\quad \left. + \tilde{f}_2(W_{t_1},\dots,W_{t_n})\frac{\partial}{\partial x_i}\tilde{f}_1(W_{t_1},\dots,x_i,\dots,W_{t_n})\bigg|_{x_i=W_{t_i}} \right) \mathbf{1}_{t<t_i}\end{aligned}$$

where we are using the product rule for the partial derivative operator. Concluding,

$$\begin{aligned}
D_t(XY) &= \tilde{f}_1(W_{t_1}, \ldots, W_{t_n}) \sum_{i=1}^{n} \left(\frac{\partial}{\partial x_i} \tilde{f}_2(W_{t_1}, \ldots, x_i, \ldots, W_{t_n}) \Big|_{x_i = W_{t_i}} \right) \mathbf{1}_{t<t_i} \\
&\quad + \tilde{f}_2(W_{t_1}, \ldots, W_{t_n}) \sum_{i=1}^{n} \left(\frac{\partial}{\partial x_i} \tilde{f}_1(W_{t_1}, \ldots, x_i, \ldots, W_{t_n}) \Big|_{x_i = W_{t_i}} \right) \mathbf{1}_{t<t_i} \\
&= X D_t(Y) + Y D_t(X).
\end{aligned}$$

This prove the product rule for D. $\square$

2.2 The Skorohod Integral

Our goal in this entire chapter is to develop an integration–by–parts equation like

$$\mathbb{E}\left[f'(X)\right] = \mathbb{E}\left[f(X)H\right]$$

for some quantity H. In this section, we develop the *Skorohod integral* because the formulation of H that we will find to complete the above integration–by–parts formula will require its use.

The Skorohod integral is denoted $\delta(.)$ and operates on processes $u = \{u_t\}$ defined on $(\Omega, \mathcal{F}, P)$. In particular, it is defined as the adjoint of the Malliavin derivative D. That is, given an arbitrary random variable F and a process u in a certain domain,[1]

$$\mathbb{E}\left[F\delta(u)\right] = \mathbb{E}\left[\int_0^T (D_t F) u_t \, \mathrm{d}t\right]. \tag{2.3}$$

We will refer to this relationship as the *duality principle*. Note that in some literature, such as [16], the notation $D^*(u)$ is used in place of $\delta(u)$ to stress this relationship with D. Heuristically, δ is a stochastic analogue to the anti–derivative of the standard differential calculus, and indeed we call $\delta(u)$ the Skorohod integral of the process u_t for that reason. This relationship between δ and the integral calculus goes far beyond heuristics, as the next proposition indicates.

Proposition 2.2.1. *For a process u_t adapted to W_t, the Skorohod integral $\delta(u)$ is equivalent to the Itô stochastic integral $\int_0^T u_t \, \mathrm{d}W_t$.*

The full proof of proposition 2.2.1 requires considerable machinery—in particular, a rigorous construction of the Skorohod integral of a function via its Wiener-Itô chaos expansion and a demonstration that the duality principle (2.3) holds under that construction. This detail is unnecessary for and beyond the scope of this work; the interested reader can find a rigorous treatment in [20, sec. 1.3] or (more gently) in [21, sec. 2].

Example 2.2.2. Computing the Skorohod integral of the process that is identically one, i.e. $u_t \equiv 1$, can be done by computing the Itô integral of the same because a constant process is adapted to W_t for any t. Thus we have

$$\delta(1) = \int_0^T \, \mathrm{d}W_t = W_T - W_0 = W_T \tag{2.4}$$

where integral is in the Itô sense. Note that the last equality holds because $W_0 = 0$ as W is a standardized $(0, 1)$ Brownian motion.

[1] The exact domain of u, which is discussed in great detail in [20, sec. 1.3], is not important for our purposes here.

Example 2.2.3. Suppose $S_t = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$ where W_t is a Wiener process. We shall encounter this process in the beginning of chapter 3—and indeed we will need the Skorohod integral $\delta(S_.)$—so we shall perform the computation now. Because S_t is adapted to W_t, we have $\delta(S_.) = \int_0^T S_t \, \mathrm{d}W_t$ where the integral is in the Itô sense.

We can use Itô's celebrated lemma to perform this computation. Itô's lemma tells us that if W_t is a Wiener process and X_t is an Itô drift–diffusion process with respect to that process governed by the stochastic differential equation

$$\mathrm{d}X_t = \tau_t \, \mathrm{d}W_t + \mu_t \, \mathrm{d}t$$

then for a twice differentiable real function $f(t,x)$ we have [3, ch. 3]

$$\mathrm{d}f(t,X_t) = \frac{\partial}{\partial t} f(t,X_t) \, \mathrm{d}t + \frac{\partial}{\partial x} f(t,X_t) \, \mathrm{d}X_t + \frac{1}{2}\tau_t^2 \frac{\partial^2}{\partial x^2} f(t,X_t) \, \mathrm{d}t. \tag{2.5}$$

Suppose in the above stochastic differential equation $\tau \equiv 1$ and $\mu \equiv 0$, i.e. $X_t = W_t$, the Wiener process itself. Then putting $\tau_t = 1$ into 2.5 we get the more specific version of Itô's lemma

$$\mathrm{d}f(t,W_t) = \frac{\partial}{\partial t} f(t,W_t) \, \mathrm{d}t + \frac{\partial}{\partial x} f(t,W_t) \, \mathrm{d}W_t + \frac{1}{2} \frac{\partial^2}{\partial x^2} f(t,W_t) \, \mathrm{d}t. \tag{2.6}$$

Let $f(t,x) = \exp\left(\left(r - \frac{1}{2}\sigma^2\right)t + \sigma x\right)$. Then $f(t,W_t) = S_t$. We can easily compute that

$$\begin{aligned}
\frac{\partial}{\partial t} f(t,x) &= \left(r - \frac{1}{2}\sigma^2\right) f(t,x) \\
\frac{\partial}{\partial x} f(x) &= \sigma f(t,x) \\
\frac{\partial^2}{\partial x^2} f(x) &= \sigma^2 f(t,x)
\end{aligned}$$

so we use Ito's lemma—in particular, the version (2.6)—to assert that

$$\begin{aligned}
\mathrm{d}f(t,W_t) &= \left(r - \frac{1}{2}\sigma^2\right) f(t,W_t) \, \mathrm{d}t + \sigma f(t,W_t) \, \mathrm{d}W_t + \frac{1}{2}\sigma^2 f(t,W_t) \, \mathrm{d}t \\
\mathrm{d}S_t &= \left(r - \frac{1}{2}\sigma^2\right) S_t \, \mathrm{d}t + \sigma S_t \, \mathrm{d}W_t + \frac{1}{2}\sigma^2 S_t \, \mathrm{d}t \\
\mathrm{d}S_t &= rS_t \, \mathrm{d}t + \sigma S_t \, \mathrm{d}W_t \\
\int_0^T \mathrm{d}S_t &= \int_0^T rS_t \, \mathrm{d}t + \int_0^T \sigma S_t \, \mathrm{d}W_t \\
S_T - S_0 &= r\int_0^T S_t \, \mathrm{d}t + \sigma \int_0^T S_t \, \mathrm{d}W_t \\
\frac{1}{\sigma}\left(S_T - S_0 - r\int_0^T S_t \, \mathrm{d}t\right) &= \int_0^T S_t \, \mathrm{d}W_t.
\end{aligned}$$

We mentioned, $\delta(S_.) = \int_0^T S_t \, \mathrm{d}W_t$, so

$$\delta(S_.) = \frac{1}{\sigma}\left(S_T - S_0 - r\int_0^T S_t \, \mathrm{d}t\right). \tag{2.7}$$

As we stated at the beginning of this example, the final identity will be useful later on.

In cases like the above two examples where our processes are well–adapted to the underlying Brownian motion, we can perform computations easily using Itô integrals. When this is not the case—when we compute $\delta(X_\cdot)$ where X_t is *not* W_t–adapted—an analytic solution to the Skorohod integral can be more difficult to find. The following relationship is often helpful.

Proposition 2.2.4. *Suppose we have a random variable F that admits a Malliavin derivative DF and a process $\{u_t\}$ that is Skorohod integrable. It need not be the case that u_t is W_t–adapted. Then*

$$\delta(Fu) = F\delta(u) - \int_0^T (D_tF)u_t \text{ d}t. \tag{2.8}$$

Proof. Following the suggestion in [16], we will make use of the duality principle (2.3) established above to prove this proposition. For any random variable $G \in \mathcal{S}$, we have by (2.3)

$$\mathbb{E}\left[G\delta(Fu)\right] = \mathbb{E}\left[\int_0^T (D_tG)Fu_t \text{ d}t\right].$$

We can now use the product rule for the D operator: since we know $D_t(FG) = FD_tG + GD_tF$, we may replace FD_tG in the integrand on the right with $D_t(FG) - GD_tF$, which gives us

$$\begin{aligned}\mathbb{E}\left[G\delta(Fu)\right] &= \mathbb{E}\left[\int_0^T (D_t(FG) - GD_tF)u_t \text{ d}t\right] = \mathbb{E}\left[\int_0^T D_t(FG)u_t \text{ d}t - \int_0^T G(D_tF)u_t \text{ d}t\right]\\ &= \mathbb{E}\left[\int_0^T D_t(FG)u_t \text{ d}t\right] - \mathbb{E}\left[G\int_0^T (D_tF)u_t \text{ d}t\right]\\ &= \mathbb{E}\left[FG\delta(u)\right] - \mathbb{E}\left[G\int_0^T (D_tF)u_t \text{ d}t\right] = \mathbb{E}\left[G\left(F\delta(u) - \int_0^T (D_tF)u_t \text{ d}t\right)\right]\end{aligned}$$

where we have applied the duality principle (2.3) again between the second and third lines. Because the above relation is true for any random variable G (of bounded expectation), we may drop the expectation operators and write

$$\delta(Fu) = F\delta(u) - \int_0^T (D_tF)u_t \text{ d}t$$

almost surely. □

Example 2.2.5. The relationship we just proved lets us calculate $\delta(W_T)$. Note that the random variable W_T is not adapted to the filtration generated by W_t: at any time $t < T$, W_T is not a measurable function. Thus in this case, the Skorohod integral $\delta(W_T)$ is not an Itô integral; nonetheless, we are able to perform the computation by letting $u \equiv 1$ and $F = W_T$. Then $\delta(F)$ is precisely the quantity on the left hand side of (2.8) and we have

$$\begin{aligned}\delta(W_T) &= W_T\delta(1) - \int_0^T (D_tW_T) \text{ d}t\\ &= W_T^2 - \int_0^T \mathbf{1}_{t<T} \text{ d}t = W_T^2 - \int_0^T \text{ d}t\\ &= W_T^2 - T \end{aligned}\tag{2.9}$$

where we have made use of our earlier computations in (2.4) and (2.2).

Incidentally, we can confirm that this result for $\delta(W_T)$ agrees with the duality principle (2.3), which asserts that for a random variable F and a process u, $\mathbb{E}\left[F\delta(u)\right] = \mathbb{E}\left[\int_0^T (D_t F) u_t \,\mathrm{d}t\right]$. Here, we could let $F = W_T$ and $u_t = 1$ for all t. Then

$$\mathbb{E}\left[F\delta(u)\right] = \mathbb{E}\left[W_T \delta(1)\right] = \mathbb{E}\left[W_T^2\right]$$

using our earlier calculation that $\delta(1) = W_T$. Of course, W is a Wiener process, so $\mathbb{E}\left[W_T\right] = 0$ and $\operatorname{var}\left[W_T\right] = T$. Then $\mathbb{E}\left[W_T^2\right] = \operatorname{var}\left[W_T\right] - \mathbb{E}\left[W_T\right]^2 = \operatorname{var}\left[W_T\right] = T$. On the other hand,

$$\begin{aligned}\mathbb{E}\left[\int_0^T (D_t F) u \,\mathrm{d}t\right] &= \mathbb{E}\left[\int_0^T D_t W_T \,\mathrm{d}t\right] \\ &= \mathbb{E}\left[\int_0^T \mathbf{1}_{t<T} \,\mathrm{d}t\right] = \mathbb{E}\left[\int_0^T \,\mathrm{d}t\right] = T,\end{aligned}$$

so we confirm that our calculation of $\delta(1) = W_T$ accords with the duality principle and the derived relationship (2.8).

We will be using the Skorohod integral to Because δ coincides with the Itô integral for many interesting processes and because we know how to compute many Itô integrals, the duality principle (2.3) gives us the opportunity to evaluate some stochastic problems analytically. Even when In particular, we have established a rule that lets us exchange a stochastic derivative with respect to a Brownian motion for an integral. In the next sections, we will use this principle to make explicit an integration–by–parts formula like the one we saw at the end of chapter 1.

2.3 Integration–by–Parts

Recalling the discussion in chapter 1, we saw that Broadie and Glasserman's "likelihood method" [5] could make Monte Carlo simulation more efficient in certain cases. That technique required us to know the density function $p_{X,\lambda}$ of the random variable X. In this section, we use the rules of Malliavin calculus as presented above to derive an integration–by–parts formula where the distribution function is not explicitly known but where the random quantities are smooth functions of a Brownian motion, which is to say that they are in our space $\mathcal{S}$ and admit Malliavin derivatives.

The following proposition is due to [16], but they state it overly specifically. Here, we choose to leave things in more general terms and provide two corollaries with specifics. The first, which is marked below as (2.10), is stated in [16]; the second, which is marked below as (2.11), does not appear in the literature. In general, many different such equations could be derived.

Proposition 2.3.1. *Suppose we have a real differentiable function f and two arbitrary random variables X and Y in the space $\mathcal{S}$. Then*

$$\mathbb{E}\left[f'(X)Y\right] = \mathbb{E}\left[f(X)\delta\left(\frac{Y h_{.}}{\int_0^T h_v D_v X \,\mathrm{d}v}\right)\right],$$

where $\delta(.)$ and $D_{.}$ represent the Skorohod integral and Malliavin derivative operators respectively and $h = \{h_s\}$ is an arbitrary process.

Proof. We define a new random variable $Z = f(X)$ for the real function f. We saw in proposition 2.1.2 that the chain rule applies to our Malliavin derivative D, so we have

$$D_s Z = f'(X) D_s X.$$

Take an arbitrary process $h = \{h_s\}$. We multiply both sides of the above equation by the product Yh_s, giving us

$$\begin{aligned}
(Yh_s) D_s Z &= (Yh_s) f'(X) D_s X \\
\int_0^T Y h_s D_s Z \, \mathrm{d}s &= \int_0^T Y h_s f'(X) D_s X \, \mathrm{d}s \\
\int_0^T Y h_s D_s Z \, \mathrm{d}s &= f'(X) Y \int_0^T h_s D_s X \, \mathrm{d}s. \\
f'(X) Y &= \frac{\int_0^T Y h_s D_s Z \, \mathrm{d}s}{\int_0^T h_s D_s X \, \mathrm{d}s} \\
&= \int_0^T (D_s Z) \frac{Y h_s}{\int_0^T h_v D_v X \, \mathrm{d}v} \, \mathrm{d}s.
\end{aligned}$$

Now, letting

$$u_s = \frac{Y h_s}{\int_0^T h_v D_v X \, \mathrm{d}v},$$

we have $f'(X)Y = \int_0^T (D_s Z) u_s \, \mathrm{d}s$. Finally,

$$\mathbb{E}\left[f'(X)Y\right] = \underbrace{\mathbb{E}\left[\int_0^T D_s Z u_s \, \mathrm{d}s\right] = \mathbb{E}\left[Z\delta(u)\right]}_{\text{Duality principle (2.3)}} = \mathbb{E}\left[f(X)\delta(u)\right],$$

which proves the theorem. □

Corollary 2.3.2. *Again, with $X, Y \in \mathcal{S}$ and f sufficiently smooth,*

$$\mathbb{E}\left[f'(X)Y\right] = \mathbb{E}\left[f(X)\delta\left(\frac{Y}{\int_0^T D_v X \, \mathrm{d}v}\right)\right] \tag{2.10}$$

Proof. The relationship expressed in proposition 2.3.1 is true for any process h; here we let $h \equiv 1$ identically. □

This integration–by–parts rule, so named because it allows us to replace differentiation with Skorohod integration, is a cousin of Broadie and Glasserman's likelihood method. Accordingly, it can make Monte Carlo simulation a more effective and precise tool even when the joint density of X and Y are unknown. We shall see applications of formula (2.10) in the next chapter.

One limitation of (2.10) is that it requires the calculation of $\int_0^T D_v X \, \mathrm{d}v$. This fails to be useful if no closed form exists for the anti–derivative of $D_v X$. The good news is that the formula expressed in proposition 2.3.1 is true for any process h, and by giving ourselves the freedom to choose the best h for the problem at hand, we can alleviate the difficulty of calculating the integral.

In many cases, it easier to differentiate a function than to find a closed form for its anti–derivative; indeed, this is the motivation for the integration–by–parts in the standard calculation. We can use this fact in the case at hand. Suppose that for the random variable X the Malliavin derivative $D_v X$ had a well-defined derivative $\frac{\partial}{\partial v} D_v X$ for all v. Then we could use proposition 2.3.1 with $h_v = \frac{\partial}{\partial v} D_v X$ to develop another integration–by–parts formula as follows.

Corollary 2.3.3. *If $X, Y \in \mathcal{S}$, f is sufficiently smooth, $D_v X$ is differentiable with respect to v for all $v \in [0,T]$, and $\int_0^T \left(\frac{\partial}{\partial v} D_v X\right) D_v X \, \mathrm{d}v \neq 0$. then*

$$\mathbb{E}\left[f'(X)Y\right] = \mathbb{E}\left[f(X)\delta\left(\frac{2Y\frac{\partial}{\partial s}D_s X}{(D_T X)^2 - (D_0 X)^2}\right)\right]. \tag{2.11}$$

Proof. As mentioned above, we use $h = \frac{\partial}{\partial x} D_v X$ in proposition 2.3.1. This gives us $\mathbb{E}\left[f'(X)Y\right] = \mathbb{E}\left[f(X)\delta(u)\right]$ with

$$u_s = \frac{Y\left(\frac{\partial}{\partial s} D_s X\right)}{\int_0^T (\frac{\partial}{\partial v} D_v X) D_v X \, \mathrm{d}v}.$$

Of course, this only makes sense if the denominator is non-zero, as required by the final assumption in the statement of the corollary. Now the integrand in the denominator has the form $\int g(v)(g'(v)dv)$, which can easily be evaluated by the standard methods of calculus; in particular, we get

$$u_s = \frac{Y\frac{\partial}{\partial s} D_s X}{\frac{1}{2}(D_v X)^2|_{v=0}^T}.$$

This directly leads to our second integration–by–parts rule (2.11). $\square$

Just as the first integration–by–parts rule (2.10) was limited by the need to compute $\int_0^T D_v X \, \mathrm{d}v$, this new rule is limited by the requirement that $\int_0^T \left(\frac{\partial}{\partial s} D_v X\right) D_v X \, \mathrm{d}v \neq 0$. That requirement is not met if the Malliavin derivative $D_v X$ is a constant process, i.e. $\frac{\partial}{\partial v} D_v X = 0$, in which case the integrand is zero.

CHAPTER 3

Malliavin Calculus for Sensitivities

In the past two chapters, we have discussed strategies for using Monte Carlo techniques to compute

$$\frac{\partial}{\partial \lambda} \mathbb{E}\left[X \mid \lambda_0\right].$$

In this chapter, we will make X and λ concrete with a number of problems from mathematical finance. Finance is rich with applications demanding the computation of sensitivities, and the models used are often too complex for analytic solution. Practitioners rely on Monte Carlo methods, and the functions they are analyzing often have discontinuities and other characteristics that we have seen make standard Monte Carlo methods ineffectual. We will apply our Malliavin integration–by–parts formulas like (2.10) and (2.11) to attack these problems.

This chapter begins with a brief introduction to mathematical finance. We try to introduce only those ideas necessary for the uninitiated reader to understand why one might want to work on the problems we encounter, and we generally offer these ideas without rigorous proof. Benth [3] gives a much more complete survey of the basics, and we will reference him along the way. We next use our Malliavin integration–by–parts formulas to develop *Malliavin estimators* for quantities known as the "greeks," the sensitivities of the prices of some financial instruments with respect to various parameters of the models used. In the third section, we perform extensive testing with a Monte Carlo simulator to compare our Malliavin estimators to the various other estimators we saw in chapter 1. We conclude the chapter by offering various extensions of the techniques used to develop Malliavin estimators for Monte Carlo approximations in other problems.

3.1 Introduction to Options and Mathematical Finance

An *option* is an agreement between two parties, the *writer* and the *holder*. In the simplest form, the holder pays the writer some amount of money up front in exchange for the opportunity—but not the obligation—to purchase a specified item from the writer for a predetermined price on a predetermined date. This type of a contract might be valuable to an airline that knew it could not afford a surge in the price of oil. By finding a willing writer, the company could, for example, purchase an option giving it the right to buy some number of barrels of oil for \$60, even if the price went up to \$100. The option is like an insurance contract for the airline: uncertain about the future, they pay a fee to the writer up front in exchange for protection if oil surges in price. Of course, if the market rate remained below \$60, the airline would let the options expire without exercising them, much like hurricane insurance that goes "unused" when there is no disaster.

The date specified in an option contract is typically referred to as the *maturity*—denoted T—and the price specified is typically referred to as the *strike*—denoted K. We assume that there is an active market for the item in question and denote the market price at maturity S_T. A rational holder will *exercise* the option if $K \le S_T$ (i.e. he buys the item below market price and can either keep it or sell it on the market and pocket $(S_T - K)$ dollars) and will ignore the option if $K > S_T$ (i.e. he would save by tearing up the option and buying on the open market). Often, these contracts are settled with cash instead of physical delivery, which means that the writer and holder only need to specify a *payoff function* $\phi(S_T)$ that specifies the amount of money the writer owes the holder on day T. In our simple example, that payoff would be

$$\phi(S_T) = (S_T - K)^+ = \max(S_T - K, 0).$$

This payoff structure is known as a *vanilla* (because it is not complicated. More generally, though, ϕ could be any function the parties agree upon whose value can be exactly computed on day T. Examples include binary, barrier, look–back, and more exotic variants; see [19, ch. 1] for a good discussion. For a more thorough introduction to the options and derivatives markets from a mathematical standpoint, see [3].

Modelling Prices

One of the focuses of financial mathematics is determining how much an option is worth on the day it is written. As the payoff ϕ is a function of some random quantities, the exact value of a certain contract cannot be known until maturity T. In theory, if one were to know the distribution of ϕ, one could determine its expected value; if the writer were demanding a lower price than that value for the option, one could profit by exploiting this mis–pricing. The mechanics of such a trading strategy could be very sophisticated, but the *fundamental theorem of asset pricing* guarantees that such a strategy will exist if the current price is not the discounted expected value of the option's payout, i.e. if

$$P \neq V_0 = \mathbb{E}_Q[\phi]$$

where Q is a risk–neutral probability measure [7, 19].

In light of the above fact, understanding the risk–neutral probability distribution of ϕ is important for the successful trading of a derivative security. The uncertainty of about the payoff ϕ demands the creation of some sort of model for the evolution of the underlying price $\{S_t\}_{t\in[0,T]}$ that ϕ is ultimately based on at maturity. Ultimately, such a model can used to compute the no–arbitrage price V_0 as defined above.

One popular model for modelling stock prices and computing option values was developed by Black and Scholes (and Merton) in the early 1970s [4]. That model, bearing their name, makes the assumption that stock prices evolve according to a geometric Brownian motion, i.e. following the stochastic differential equation

$$\mathrm{d}S_t = rS_t \,\mathrm{d}t + \sigma S_t \,\mathrm{d}W_t$$

where r denotes the risk–free interest rate, σ the volatility of the stock in question, and W a Wiener process. Using the Itô calculus, this differential equation leads to a simple formula for the stock price evolution as a function of that Wiener process, namely

$$S_t = S_0 e^{\mu T + \sigma W_t}$$

where $\mu = r - \frac{1}{2}\sigma^2$ [3, p. 60–66]. Part of the popularity of this model comes from the simplicity of this expression and of the expressions for the prices of some commonly traded options.[1]

Black–Scholes has various problems—notably, it assumes constant volatility—that can be corrected for with various adjustments or entirely new models. A refinement could incorporate a dynamically changing risk–free rate r or volatility σ, which might add realism at the expense of simplicity [12, p. 101–108]. Indeed, the parameters r and σ could evolve stochastically, each governed by its own dynamics. The evolution of r can be modelled as a mean–reverting geometric Brownian motion (as in the Vasicek model) [12, p. 108]; similarly, volatility σ could be driven by a chance parameter. Some models allow the price to diffuse normally but then experience "jumps" that arrive in a Poisson process [12, sec. 3.5].

In this work, we will assume that prices follow a geometric Brownian motion as Black and Scholes did. This work is about a technique in Monte Carlo simulation, not a survey of various types of pricing models, and the geometric Brownian motion lends itself to fast and reasonably straightforward computations. It is important to note that the Malliavin estimators we develop are specific to the geometric Brownian motion; some very recent work has begun the development of similar methods for processes with jumps [6]. In section 3.5.1 at the end of this chapter, we will attempt to generalize our methods to solve problems without the assumption that prices follow a geometric Brownian motion.

Greeks

Regardless of the model used, a practitioner of mathematical finance tries to estimate

$$V_0 = \mathbb{E}_Q[\phi]$$

to identify market mis–pricings. Such an estimate can change quickly as market conditions evolve, however. In the Black–Scholes model, for example, the parameters S_0 (current price of the underlying security), r (interest rate), and σ (volatility) can all change rapidly in the enormously complicated global markets. Such changes will change V_0: it is sensitive to the parameters in the model. We refer to the sensitivity of V_0 to one of its parameters as a *greek*.

There are many different greeks expressing higher–order and cross–sensitivities to all sorts of different parameters in the models used. For the present work, we focus on three fairly simple ones:

- Δ ("delta") is defined as the first derivative of the option price V_0 with respect to the price S_0 of the underlying security.

- Γ ("gamma") is defined as the second derivative of the option price V_0 with respect to the price S_0 of the underlying security. Note that $\Gamma = \frac{\partial}{\partial S_0}\Delta$.

- $\mathcal{V}$ ("vega") is defined as the first derivative of the option price V_0 with respect to the volatility σ of the underlying security.

[1] The reader will recognize this formulation for S_t as the process of which we computed the Skorohod integral in example 2.2.3.

The first of these, Δ, is of particular importance because it helps define the optimal continuous time trading strategy in the Black–Scholes model [3]. The first derivative of Δ, known as Γ, is important because of its relation to Δ. $\mathcal{V}$ is useful because it informs the holder of an option about his risks if the market becomes volatile overnight. These three are not the only important greeks—sensitivity to the interest rate and to the passage of time are useful to practitioners—but their computation offers a wide variety of problems. As the reader may have guessed, we will be using the Malliavin calculus to develop estimators for Δ, Γ, and $\mathcal{V}$ to help us perform these computations.

3.2 Greeks for European Options with Malliavin Calculus

A European-style option is one where the payoff at maturity is only a function of the final price of the underlying security S_T. These are the simplest sort of option contract as the value at expiration does not depend on the particular path $\{S_t\}_{t\in[0,T]}$ that the underlying price took to reach S_T.

The valuation of these options and the calculation of their sensitivities to various parameters are relatively easy next to path-dependent options. Indeed, under the geometric Brownian motion (Black–Scholes) model discussed above, analytic solutions exist for both the value V_0 and the greeks. In this section, we use the Malliavin calculus integration–by–parts rule developed in chapter 2 to create unbiased estimators for the greeks under the assumption that the price S_t follows a geometric Brownian motion. In section 3.4, we will compare the performance of these Malliavin estimators with the exact values given by the analytic solutions to the problem.

The computations of the greeks for European options that follow are explanations of the identities first derived in [9].

Proposition 3.2.1 (Δ for a European Option)**.** *Suppose we have a European–style option with payoff ϕ that follows a geometric Brownian motion $\{S_t\}_{t\in[0,T]}$ with risk–free rate r, maturity T, volatility σ, and initial condition S_0. Then the sensitivity Δ of that option's value with respect to S_0 is given by*

$$\Delta = \frac{e^{-rT}}{\sigma S_0 T}\mathbb{E}\left[\phi(S_T)W_T\right] \tag{3.1}$$

where W_T is the Wiener process that drives the geometric Brownian motion.

Proof. Calculating the Δ of a European option will be the most straightforward application of the Malliavin integration–by–parts formulas. Recalling that Δ is the sensitivity of the price V_0 with respect to the current price of the underlying S_0 and that asset prices are discounted expectations of their future values, we have

$$\begin{aligned}\Delta &= \frac{\partial}{\partial S_0}e^{-rT}\mathbb{E}\left[\phi(S_T)\right] = e^{-rT}\mathbb{E}\left[\frac{\partial}{\partial S_0}\phi(S_T)\right] \\ &= e^{-rT}\mathbb{E}\left[\phi'(S_T)\frac{\partial S_T}{\partial S_0}\right] = e^{-rT}\mathbb{E}\left[\phi'(S_T)\frac{\partial}{\partial S_0}\left(S_0 e^{\mu T+\sigma W_T}\right)\right] \\ &= e^{-rT}\mathbb{E}\left[\phi'(S_T)e^{\mu T+\sigma W_T}\right] = \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi'(S_T)S_T\right]\end{aligned}$$

where the prime denotes differentiation of ϕ with respect to its only argument (the stock price at expiration). We now notice that the expectation in the expression for Δ derived above is in a familiar form: the derivative of a real function of a random variable X multipled by another random variable Y (here $X = Y = S_T$). We can thus apply the integration–by–parts rule (2.10) to obtain

$$\Delta = \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(S_T)H\right]$$

where the random variable H is given by

$$H = \delta\left(\frac{S_T}{\int_0^T D_v S_T \,\mathrm{d}v}\right)$$

for some random process h, as specified above.

We can calculating H with the tools at our disposal. First, we make explicit the dependence of S_T on the process W_T by writing $S_T = \psi(W_T)$ where $\psi(x) = S_0 e^{\mu T + \sigma x}$. We can then use the Malliavin operator D to calculate

$$D_v S_T = D_v \psi(W_T) = \psi'(W_T) D_v W_T = \sigma\psi(W_T)\mathbf{1}_{v<T} = \sigma S_T \mathbf{1}_{v<T} \tag{3.2}$$

where we have used the chain rule for D and the fact that $\frac{\partial}{\partial x} S_0 e^{\mu T+\sigma x} = \sigma S_0 e^{\mu T + \sigma x}$.

We can then calculate the full integral in the denominator to be

$$\int_0^T D_v S_T \,\mathrm{d}v = \int_0^T \sigma S_T \mathbf{1}_{v<T} \,\mathrm{d}v = \sigma S_T T \tag{3.3}$$

because $v \leq T$ on the entire domain of integration. We conclude

$$H = \delta\left(\frac{S_T}{\sigma S_T T}\right) = \delta\left(\frac{1}{\sigma T}\right) = \frac{\delta(1)}{\sigma T}$$

we are then ready to compute Δ. Plugging in our expression for H, we have

$$\Delta = \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(S_T)H\right] = \frac{e^{-rT}}{\sigma T S_0}\mathbb{E}\left[\phi(S_T)\delta(1)\right] = \frac{e^{-rT}}{\sigma S_0 T}\mathbb{E}\left[\phi(S_T)W_T\right]$$

where we have used the fact that $\delta(1) = W_T$, as calculated in (2.4). □

Proposition 3.2.2 ($\mathcal{V}$ for a European Option)**.** *Under the same assumptions as proposition 3.2.1, the sensitivity $\mathcal{V}$ of a European–style option's value to changes in σ is given by*

$$\mathcal{V} = e^{-rT}\mathbb{E}\left[\phi(S_T)\left(\frac{W_T^2 - T}{\sigma T} - W_T\right)\right] \tag{3.4}$$

Proof. Recall that $\mathcal{V}$ is the sensitivity of an option's price to the volativity of the underlying asset. Under the Black-Scholes model that we have been employing, volatility enters model with the parameter σ in the expression for the underlying security's price

$$S_T = S_0 e^{\mu T + \sigma W_T} = S_0 e^{(r - \frac{\sigma^2}{2})T + \sigma W_T}$$

(recalling that $\mu = r - \frac{\sigma^2}{2}$).

We begin with calculations similar to the ones we performed in calculating Δ:

$$\begin{aligned}\mathcal{V} = \frac{\partial V_0}{\partial \sigma} = \frac{\partial}{\partial \sigma}\mathbb{E}\left[e^{-rT}\phi(S_T)\right] = e^{-rT}\mathbb{E}\left[\phi'(S_T)\frac{\partial}{\partial \sigma}\left(S_0 e^{(r-\frac{\sigma^2}{2})T+\sigma W_T}\right)\right] \\ = e^{-rT}\mathbb{E}\left[\phi'(S_T)\left(S_0 e^{\mu t+\sigma W_T}\right)(W_T-\sigma T)\right] = e^{-rT}\mathbb{E}\left[\phi'(S_T)S_T\left(W_T-\sigma T\right)\right]\end{aligned}$$

We can now apply our integration–by–parts formula (2.10) with $X = S_T$ and $Y = S_T(W_T - \sigma T)$. This gives us

$$\mathcal{V} = e^{-rT}\mathbb{E}\left[\phi(S_T)\delta\left(\frac{S_T(W_T-\sigma T)}{\int_0^T D_v S_T \, \mathrm{d}v}\right)\right].$$

Using our earlier calculation (3.3) of $\int_0^T D_v S_T \, \mathrm{d}v = \sigma S_T T$ when we computed Δ for a European option, we obtain

$$\begin{aligned}\mathcal{V} &= e^{-rT}\mathbb{E}\left[\phi(S_T)\delta\left(\frac{W_T-\sigma T}{\sigma T}\right)\right] \\ &= e^{-rT}\mathbb{E}\left[\phi(S_T)\left(\frac{\delta(W_T)}{\sigma T}-\delta(1)\right)\right]\end{aligned}$$

where we have used the linearity of the δ operator. Now, we have already calculated

$$\begin{aligned}\delta(W_T) &= W_T^2 - T \\ \delta(1) &= W_T\end{aligned}$$

in (2.9) and (2.4) respectively, and so the final expression for $\mathcal{V}$ follows immediately:

$$\mathcal{V} = e^{-rT}\mathbb{E}\left[\phi(S_T)\left(\frac{W_T^2-T}{\sigma T}-W_T\right)\right]$$ □

Proposition 3.2.3 (Γ for a European Option)**.** *Under the same assumptions as proposition 3.2.1, the sensitivity Γ of a European–style option's value to second order changes in S_0 is given by*

$$\Gamma = \frac{e^{-rT}}{S_0^2\sigma T}\mathbb{E}\left[\phi(S_T)\left(\frac{W_T^2-T}{\sigma T}-W_T\right)\right]. \tag{3.5}$$

Proof. The calculation of Γ, the second derivative of the option's value with respect to the underlying stock price, differs from the above slightly in that it requires us to perform integration–by–parts twice. Though this requires some care, it is not substantially more difficult than the calculations of Δ and $\mathcal{V}$ for European Options. In particular we perform normal manipulations to determine that

$$\begin{aligned}\Gamma &= \frac{\partial^2}{\partial S_0^2}V_0 = \frac{\partial^2}{\partial S_0^2}e^{-rT}\mathbb{E}\left[\phi(S_T)\right] \\ &= e^{-rT}\mathbb{E}\left[\frac{\partial}{\partial S_0}\phi'(S_T)(e^{\mu T+\sigma W_T})\right] = e^{-rT}\mathbb{E}\left[\phi''(S_T)(e^{\mu T+\sigma W_T})^2\right] \\ &= e^{-rT}\mathbb{E}\left[\phi''(S_T)\left(\frac{S_T}{S_0}\right)^2\right] = \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi''(S_T)S_T^2\right].\end{aligned}$$

We now apply our integration–by–parts formula (2.10) with $X = S_T$ and $Y = S_T^2$, yielding

$$\begin{aligned}\Gamma &= \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi'(S_T)\delta\left(\frac{S_T^2}{\int_0^T D_v S_T \text{ d}v}\right)\right] \\ &= \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi'(S_T)\delta\left(\frac{S_T^2}{\sigma S_T T}\right)\right] \\ &= \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi'(S_T)\frac{1}{\sigma T}\delta(S_T)\right]\end{aligned}$$

Calculating $\delta(S_T)$ is relatively easy with the transformation (2.8), setting $F = S_T$ and $u \equiv 1$ identically. This gives us

$$\int_0^T S_T \text{ d}W_T = S_T \int_0^T \text{ d}W_t - \int_0^T D_t S_T \text{ d}t$$
$$\delta(S_T) = S_T\delta(1) - \sigma S_T T = S_T(W_T - \sigma T)$$

Where the last line uses the calculations already performed in (2.4) and (3.3). Thus

$$\begin{aligned}\Gamma &= \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi'(S_T)S_T\left(\frac{W_T}{\sigma T} - 1\right)\right] \\ &= \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi(S_T)\delta\left(\frac{S_T(\frac{W_T}{\sigma T} - 1)}{\int_0^T D_v S_T \text{ d}v}\right)\right]\end{aligned}$$

applying the integration–by–parts rule (2.10) for the second time with $X = S_T$ and $Y = S_T(\frac{W_T}{\sigma T} - 1)$. The denominator yet again simplifies to $\sigma S_T T$, and we are left with

$$\Gamma = \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi(S_T)\delta\left(\frac{S_T(\frac{W_T}{\sigma T} - 1)}{\sigma S_T T}\right)\right] = \frac{e^{-rT}}{S_0^2 \sigma T}\mathbb{E}\left[\phi(S_T)\delta\left(\frac{W_T - \sigma T}{\sigma T}\right)\right]$$

We saw in the calculation of $\mathcal{V}$ that $\delta\left(\frac{W_T - \sigma T}{\sigma T}\right) = \frac{W_T^2 - T}{\sigma T} - W_T$, so our final result is

$$\Gamma = \frac{e^{-rT}}{S_0^2 \sigma T}\mathbb{E}\left[\phi(S_T)\left(\frac{W_T^2 - T}{\sigma T} - W_T\right)\right],$$

completing the proof. □

Incidentally, (3.5) and (3.4) confirm a (well-known) relationship between Γ and $\mathcal{V}$ under geometric Brownian motion, namely that

$$\Gamma = \frac{\mathcal{V}}{S_0^2 \sigma T}.$$

We have now developed the estimators that were first found in 1999 in [9] for Δ, $\mathcal{V}$, and Γ using our integration–by–parts rule (2.10). It is worth noting that the work done in [9] used a different integration–by–parts rule; ours is more like the one used in [16], which makes the algebra much cleaner and produces an identical result.

The reader will recall that we actually developed two different integration–by–parts rules—the other being (2.11)—that could be used to deal with this class of problem. In this case, however, the latter transformation is not valid: We recall the requirement in the statement of corollary 2.3 that $\int_0^T \left(\frac{\partial}{\partial v} D_v X\right) D_v X \, \mathrm{d}v \neq 0$ and the ensuing discussion of how this requirement is not met when $D_v X$ is constant in v. Every time we used an integration–by–parts rule in this section (i.e. with greeks for European options), we had $X = S_T$, and we saw in (3.2) that $D_v S_T = \sigma S_T \mathbf{1}_{v<T} \equiv \sigma S_T$ as $v < T$ on our entire domain. Thus our second integration–by–parts rule (2.11) is not applicable as $\frac{\partial}{\partial v} D_v S_T \equiv 0$.

3.3 Greeks for Asian Options with Malliavin Calculus

In a European–style contract, the payoff is entirely a function of the price on a single day. This makes such contracts vulnerable to price manipulation: the holder of a large number of call options might take some action (illegally, perhaps) to make the price artificially surge on the day in question to make a windfall profit. Asian–style options make this sort of cheating more difficult by computing the payoff ϕ as a function of the price S_t on multiple days, a sort of averaging.

In simple constructions, the average might be taken arithmetically over a finite set of times. For example, the payoff function might be $\phi\left(S_{T/4}, S_{T/2}, S_{3T/4}, S_T\right)$. Malliavin estimators similar to the ones we found for European options can easily be derived in this case, though the computation are long and reveal little of interest. A more sophisticated Asian option might compute the average continuously, i.e. using a payoff function $\phi(\bar{S}_T)$ where $\bar{S}_T = \int_0^T S_t \, \mathrm{d}t$. A simple example could be the continuous Asian call, where the payoff ϕ takes the form $\left(\frac{1}{T}\bar{S}_T - K\right)^+$. Whereas our computations for European options were moot in that direct solutions with analytic methods existed, even the simplest arithmetic continuous Asian options require some sort of numerical method for computing greeks.

In this section, we develop Malliavin estimators for the Δ and Γ of an Asian–style option. Variations on the Δ computation can be found in [1] or [20, ch. 6] (there are several different versions of these computations owing to the use of variations of the integration–by–parts formulas). The Γ we find is newly developed and cannot be found in the literature. Both computations make use of our second Malliavin integration–by–parts theorem (2.11).

Proposition 3.3.1 (Δ for an Asian Option)**.** *Suppose we have an Asian–style option with path–dependent payoff ϕ that follows a geometric Brownian motion $\{S_t\}_{t\in[0,T]}$ with risk–free rate r, maturity T, volatility σ, and initial condition S_0. Then the sensitivity Δ of that option's value with respect to S_0 is given by*

$$\Delta = \frac{e^{-rT}}{S_0\sigma^2}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{2(S_T - S_0)}{\bar{S}_T} + \sigma^2 - 2r\right)\right]. \tag{3.6}$$

Proof. We begin as we did for European-style options: the value of a contingent claim is once again a discounted expectation of the payoff, i.e.

$$V_0 = e^{-rT}\mathbb{E}\left[\phi(\bar{S}_T)\right].$$

We then differentiate this with respect to the appropriate parameter to determine the greek. For

Δ, we have

$$\begin{aligned}\Delta = \frac{\partial}{\partial S_0}V_0 &= e^{-rT}\mathbb{E}\left[\frac{\partial}{\partial S_0}\phi(\bar{S}_T)\right] \\ &= e^{-rT}\mathbb{E}\left[\phi'(\bar{S}_T)\frac{\partial}{\partial S_0}\bar{S}_T\right] = e^{-rT}\mathbb{E}\left[\phi'(\bar{S}_T)\int_0^T \frac{\partial}{\partial S_0}S_T \text{ d}t\right] \\ &= \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi'(\bar{S}_T)\int_0^T S_T \text{ d}t\right] = \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi'(\bar{S}_T)\bar{S}_T\right] \end{aligned} \tag{3.7}$$

where we have again used the fact that, if the price S_t follows the Black-Scholes model,

$$\frac{\partial}{\partial S_0}S_T = \frac{\partial}{\partial S_0}S_0e^{\mu T+\sigma W_T} = \frac{S_T}{S_0}.$$

We now need to use an integration–by–parts formula to change $\mathbb{E}\left[\phi'(\bar{S}_T)\bar{S}_T\right]$ into $\mathbb{E}\left[\phi(\bar{S}_T)H\right]$ for some random quantity H. In the last section, we used the integration–by–parts formula (2.10) with success for European options, but here we are not so lucky: applying (2.10) here would require us to compute $\int_0^T D_v\bar{S}_T \text{ d}v$. We can see that

$$D_s\bar{S}_T = D_s\int_0^T S_t \text{ d}t = \int_0^T D_sS_t \text{ d}t = \int_0^T \sigma S_t\mathbf{1}_{s<t} \text{ d}t = \sigma\int_s^T S_t \text{ d}t, \tag{3.8}$$

which means we would need to compute $\int_0^T\int_v^T S_t \text{ d}t \text{ d}v$; on top of that, we would be forced to compute the Skorohod integral of the result of that integral. Though this approach might be feasible, we have another tool at our disposal: the second integration–by–parts formula (2.11). That formula tells us that

$$\mathbb{E}\left[f'(X)Y\right] = \mathbb{E}\left[f(X)\delta\left(\frac{2Y\frac{\partial}{\partial \cdot}D_\cdot X}{(D_TX)^2-(D_0X)^2}\right)\right]$$

if D_vX is differentiable with respect to v and $\int_0^T\left(\frac{\partial}{\partial v}D_vS_T\right)D_vS_T \text{ d}v \neq 0$. In light of (3.7), it is natural to use $X = \bar{S}_T$. We therefore need to ensure that $\frac{\partial}{\partial s}D_vX = \frac{\partial}{\partial v}D_v\bar{S}_T$ exists and is not orthogonal to D_vS_T. We already calculated D_sS_T above; we can easily differentiate that expression with respect to s using the fundamental theorem of calculus, which gives us

$$\frac{\partial}{\partial s}D_s\bar{S}_T = \frac{\partial}{\partial s}\sigma\int_s^T S_r \text{ d}r = -\sigma\frac{\partial}{\partial s}\int_T^s S_r \text{ d}r = -\sigma S_s. \tag{3.9}$$

We now can confirm that $\int_0^T\left(\frac{\partial}{\partial v}D_vS_T\right)D_vS_T \text{ d}v \neq 0$: (3.8) is positive for all $s < T$ and (3.9) is negative for all $s < T$, so their product is negative for all $s < T$ and thus the integral of their product is definitely non–zero. Having confirmed that our integration–by–parts formula (2.11) is applicable (and having conveniently computed $\frac{\partial}{\partial s}D_s\bar{S}_T$ along the way), we continue with our calculation.

The other quantities we need are $D_T\bar{S}_T$ and $D_0\bar{S}_T$; these are easy because we have already calculated that $D_s\bar{S}_T = \sigma\int_s^T S_t \text{ d}t$. We thus have

$$D_T\bar{S}_T = \sigma\int_T^T S_t \text{ d}t = 0$$

$$D_0\bar{S}_T = \sigma\int_0^T S_t \text{ d}t = \sigma\bar{S}_T$$

Putting everything together, we can assert that

$$\begin{aligned}\Delta &= \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(\bar{S}_T)\delta\left(\frac{2\bar{S}_T(-S_\cdot)}{-(\sigma\bar{S}_T)^2}\right)\right]\\ &= \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(\bar{S}_T)\delta\left(\frac{2S_\cdot}{\sigma\bar{S}_T}\right)\right]\end{aligned}$$

To calculate the Skorohod integral $\delta(\frac{2S_\cdot}{\sigma\bar{S}_T})$, we will use the helpful relationship (2.8) with $u = S_\cdot$ and $F = \frac{1}{\bar{S}_T}$. We have

$$\begin{aligned}\delta\left(\frac{2S_\cdot}{\sigma\bar{S}_T}\right) &= \frac{2}{\sigma}\delta\left(S_\cdot\frac{1}{\bar{S}_T}\right) = \frac{2}{\sigma}\left(\delta(S_\cdot)\frac{1}{\bar{S}_T} - \int_0^T S_t D_t\left(\frac{1}{\bar{S}_T}\right)\,\mathrm{d}t\right)\\ &= \frac{2}{\sigma}\left(\frac{\delta(S_\cdot)}{\bar{S}_T} - \int_0^T S_t\left(\frac{-D_t\bar{S}_T}{\bar{S}_T^2}\right)\,\mathrm{d}t\right)\\ &= \frac{2}{\sigma}\left(\frac{\delta(S_\cdot)}{\bar{S}_T} + \frac{\int_0^T S_t D_t\bar{S}_T\,\mathrm{d}t}{\bar{S}_T^2}\right)\end{aligned}$$

As for the integral in the numerator, we have

$$\begin{aligned}\int_0^T S_t D_t\bar{S}_T\,\mathrm{d}t &= \int_0^T S_t\left(\sigma\int_t^T S_r\,\mathrm{d}r\,\mathrm{d}t\right) = -\sigma\int_0^T(-S_t)\int_t^T S_r\,\mathrm{d}r\,\mathrm{d}t\\ &= -\frac{\sigma}{2}\left(\int_t^T S_r\,\mathrm{d}r\right)^2\Bigg|_{t=0}^{T} = -\frac{\sigma}{2}\left(\left(\int_T^T S_r\,\mathrm{d}r\right) - \left(\int_0^T S_r\,\mathrm{d}r\right)^2\right) = \frac{\sigma}{2}\bar{S}_T^2.\end{aligned}$$

Our expression for Δ is now

$$\Delta = \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(\bar{S}_T)\frac{2}{\sigma}\left(\frac{\delta(S_\cdot)}{\bar{S}_T} + \frac{\sigma\bar{S}_T^2}{2\bar{S}_T^2}\right)\right],$$

which requires us to compute the Skorohod integral $\delta(S_\cdot)$. Fortunately, we computed this value in equation (2.7) in example 2.2.3 on page 18 to be $\frac{1}{\sigma}\left(S_T - S_0 - r\bar{S}_T\right)$. Incorporating that result here,

$$\begin{aligned}\Delta &= \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(\bar{S}_T)\frac{2}{\sigma}\left(\frac{\delta(S_\cdot)}{\bar{S}_T} + \frac{\sigma\bar{S}_T^2}{2\bar{S}_T^2}\right)\right]\\ &= \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{2}{\sigma^2}\frac{S_T - S_0 - r\bar{S}_T}{\bar{S}_T} + 1\right)\right]\\ &= \frac{e^{-rT}}{S_0\sigma^2}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{2(S_T - S_0)}{\bar{S}_T} + \sigma^2 - 2r\right)\right].\end{aligned}$$ □

Proposition 3.3.2 (Calculating Γ for an Asian Option)**.** *Under the same assumptions as proposition 3.3.1, the sensitivity Γ of an Asian–style option's value to changes in σ is given by*

$$\Gamma = \frac{4e^{-rT}}{\sigma^3 S_0^2}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{(S_T - S_0)^2 - (S_T - S_0)\,r\bar{S}_T}{\sigma\bar{S}_T^2} - \frac{\sigma S_0}{\bar{S}_T}\right)\right] - \frac{2r}{\sigma^2 S_0}\Delta \tag{3.10}$$

Proof. As was the case with the European-style option, the calculation of Γ is more sophisticated than that of Δ. We begin as always by differentiating the expression for the price of a contingent claim and moving the derivative within the expectation operator, as allowed by the Liebnitz rule:

$$\begin{aligned}\Gamma = \frac{\partial^2}{\partial S_0^2} V_0 = e^{-rT}\mathbb{E}\left[\frac{\partial^2}{\partial S_0^2}\phi(\bar{S}_T)\right] &= e^{-rT}\mathbb{E}\left[\phi''(\bar{S}_T)\frac{\partial^2}{\partial S_0^2}\bar{S}_T\right] \\ = \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi''(\bar{S}_T)\bar{S}_T^2\right] &= \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi'(\bar{S}_T)\delta(u)\right]\end{aligned}$$

where $u_s = \frac{2\bar{S}_T^2 \frac{\partial}{\partial s} D_s \bar{S}_T}{(D_T\bar{S}_T)^2 - (D_0\bar{S}_T)^2}$ as specified by our general integration–by–parts formula (2.11) with $X = \bar{S}_T$ and $Y = \bar{S}_T^2$. We already computed $\frac{\partial}{\partial s} D_s \bar{S}_T = -\sigma S_s$ in (3.9), so we can proceed to calculate

$$\Gamma = \frac{e^{-rT}}{S_0^2}\mathbb{E}\left[\phi'(\bar{S}_T)\delta\left(\frac{-2\bar{S}_T^2\sigma S_\cdot}{-(\sigma\bar{S}_T)^2}\right)\right] = \frac{2e^{-rT}}{\sigma S_0^2}\mathbb{E}\left[\phi'(\bar{S}_T)\delta\left(S_\cdot\right)\right].$$

Using the fact that $\delta(S_\cdot) = \frac{1}{\sigma}\left(S_T - S_0 - r\bar{S}_T\right)$, which we saw in (2.7) and the linearity of the expectation operator, we conclude that

$$\Gamma = \frac{2e^{-rT}}{\sigma^2 S_0^2}\left(\mathbb{E}\left[\phi'(\bar{S}_T)\left(S_T - S_0\right)\right] - \mathbb{E}\left[\phi'(\bar{S}_T)\left(r\bar{S}_T\right)\right]\right) \tag{3.11}$$

Recalling our calculation of Δ for an Asian option, an early step in the computation gave us $\Delta = \frac{e^{-rT}}{S_0}\mathbb{E}\left[\phi'(\bar{S}_T)\bar{S}_T\right]$, which we recognize as being very similar to the right-hand expectation above. We can indeed replace the right hand term of (3.11) with the simplification

$$\mathbb{E}\left[\phi'(\bar{S}_T)\left(r\bar{S}_T\right)\right] = \frac{S_0 r}{e^{-rT}}\Delta. \tag{3.11a}$$

As for the left-hand expectation in (3.11), we can apply our general integration–by–parts formula (2.11) a second time to remove the derivative on from the function ϕ. This gives us

$$\begin{aligned}\mathbb{E}\left[\phi'(\bar{S}_T)\left(S_T - S_0\right)\right] &= \mathbb{E}\left[\phi(\bar{S}_T)\delta\left(\frac{2(S_T - S_0)(-\sigma S_\cdot)}{-(\sigma\bar{S}_T)^2}\right)\right] \\ &= \frac{2}{\sigma}\mathbb{E}\left[\phi(\bar{S}_T)\delta\left(\frac{(S_T - S_0)S_\cdot}{\bar{S}_T^2}\right)\right]\end{aligned}$$

We can compute the Skorohod integral in this expression using the transformation law (2.8) that we have seen to be helpful many times already. In particular, we apply (2.8) with $u_\cdot = S_\cdot$ and $F = \frac{S_T - S_0}{\bar{S}_T^2}$, giving us

$$\delta\left(\frac{(S_T - S_0)S_\cdot}{\bar{S}_T^2}\right) = \frac{S_T - S_0}{\bar{S}_T^2}\delta(S_\cdot) - \int_0^T S_t D_t\left(\frac{S_T - S_0}{\bar{S}_T^2}\right)\,\mathrm{d}t$$

The left hand term of this expression takes on its simplest form when we again use the fact that $\delta(S_\cdot) = \frac{1}{\sigma}\left(S_T - S_0 - r\bar{S}_T\right)$; for the right hand term, we use the quotient rule (a direct result of the product rule, which we have proved) to calculate the Malliavin derivative

$$D_t\left(\frac{S_T - S_0}{\bar{S}_T^2}\right) = \frac{\bar{S}_T^2 D_t(S_T - S_0) - (S_T - S_0)(2\bar{S}_T)D_t\bar{S}_T}{\bar{S}_T^4}$$

Since the D operator is linear, $D_t(S_T - S_0) = D_tS_T - D_tS_0 = \sigma S_T \mathbf{1}_{t<T} - 0$, where we have used (3.2) to compute the individual Malliavin derivatives. Using the result for $D_t\bar{S}_T$ that we established in (3.8), our entire statement becomes

$$\begin{aligned} D_t\left(\frac{S_T - S_0}{\bar{S}_T^2}\right) &= \frac{\bar{S}_T^2(\sigma S_T\mathbf{1}_{t<T} - 0) - 2(S_T - S_0)\bar{S}_T\sigma\int_t^T S_r \, \mathrm{d}r}{\bar{S}_T^4} \\ &= \sigma\frac{S_T}{\bar{S}_T^2}\mathbf{1}_{t<T} - 2\sigma\frac{(S_T - S_0)}{\bar{S}_T^3}\int_t^T S_r \, \mathrm{d}r. \end{aligned}$$

We plug this result into the integral we are trying to evaluate and find that

$$\begin{aligned} \int_0^T S_tD_t\left(\frac{S_T - S_0}{\bar{S}_T^2}\right) \mathrm{d}t &= \sigma\int_0^T S_t\left(\frac{S_T}{\bar{S}_T^2}\mathbf{1}_{t<T}\right) \mathrm{d}t - 2\sigma\int_0^T S_t\left(\frac{(S_T - S_0)}{\bar{S}_T^3}\int_t^T S_r \, \mathrm{d}r\right) \mathrm{d}t \\ &= \frac{\sigma S_T}{\bar{S}_T^2}\underbrace{\int_0^T S_t \, \mathrm{d}t}_{=\bar{S}_T} - \frac{2\sigma(S_T - S_0)}{\bar{S}_T^3}\underbrace{\int_0^T S_t\left(\int_t^T S_r \, \mathrm{d}r\right) \mathrm{d}t}_{=\frac{1}{2}\bar{S}_T^2} \\ &= \frac{\sigma S_T}{\bar{S}_T} - \frac{\sigma(S_T - S_0)}{\bar{S}_T} = \frac{\sigma S_0}{\bar{S}_T} \end{aligned}$$

where the simplification $\int_0^T S_t\left(\int_t^T S_r \, \mathrm{d}r\right) = \frac{1}{2}\bar{S}_T^2$ is similar to the one performed while calculating Δ in the preceding section. We conclude that

$$\mathbb{E}\left[\phi'(\bar{S}_T)\left(S_T - S_0\right)\right] = \frac{2}{\sigma}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{S_T - S_0}{\sigma\bar{S}_T^2}\left(S_T - S_0 - r\bar{S}_T\right) - \frac{\sigma S_0}{\bar{S}_T}\right)\right] \tag{3.11b}$$

Finally, plugging (3.11b) and (3.11b) into (3.11), we come up with our final expression for Γ:

$$\begin{aligned} \Gamma &= \frac{2e^{-rT}}{\sigma^2S_0^2}\left(\frac{2}{\sigma}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{S_T - S_0}{\bar{S}_T^2}\left(\frac{1}{\sigma}\left(S_T - S_0 - r\bar{S}_T\right)\right) - \frac{\sigma S_0}{\bar{S}_T}\right)\right] - \frac{rS_0}{e^{-rT}}\Delta\right) \\ &= \frac{4e^{-rT}}{\sigma^3S_0^2}\mathbb{E}\left[\phi(\bar{S}_T)\left(\frac{(S_T - S_0)^2 - (S_T - S_0)\,r\bar{S}_T}{\sigma\bar{S}_T^2} - \frac{\sigma S_0}{\bar{S}_T}\right)\right] - \frac{2r}{\sigma^2S_0}\Delta \qquad \square \end{aligned}$$

Since this calculation uses Δ to calculate Γ, a intelligent implementation of this system would compute Δ first.

3.4 Numerical Investigation and Efficiency

In this section, we investigate the effectiveness of the Malliavin estimators that we have developed in this chapter. We use Monte Carlo methods to implement these methods and other approaches to the problem of computing sensitivities, and we compare the results. In particular, we attempt to answer two questions:

1. We have mentioned that when the price of an underlying security follows a geometric Brownian motion, exact values for greeks for European options can be computed analytically. How do the Malliavin estimates of greeks compare with those exact values? That is, are we right that the Malliavin estimators are unbiased?

2. How do the Malliavin estimators of sensitivities with other numerical methods for computing the same values? In particular, how do they compare with the approaches we discussed in chapter 1?

3.4.1 Comparison with Analytic Method for European Options

The original work of Black and Scholes [4] gives an analytic solution for the value V_0 of a European call option with a vanilla payoff $\phi(S_T) = (S_T - K)^+$. This value can be computed as a function of maturity T, underlying stock price S, strike K, risk-free interest rate r, and assumed volatility σ. In particular,

$$V_0 = S\mathcal{N}(d_1) - e^{-rT}K\mathcal{N}(d_2) \tag{3.12}$$

where $\mathcal{N}(x) = \int_{-\infty}^{x} \exp(-z^2/2)/\sqrt{2\pi}\,\mathrm{d}z$ is the cumulative density function of a normalized Gaussian distribution,

$$d_1 = \frac{\log(S/K) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\log(S/K) + (r - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$

and σ is the volatility assumed for the stock price during the period. Note the helpful identity $d_2 = d_1 - \sigma\sqrt{T}$.

We can calculate closed formulas for the greeks of a European call option by differentiating the formula for price (3.12) with respect to the parameter of interest. Calculating Δ requires us to compute

$$\begin{aligned}\Delta &= \frac{\partial}{\partial S}\left(S\mathcal{N}(d_1) - e^{-rT}K\mathcal{N}(d_2)\right) = \frac{\partial}{\partial S}\left(S\mathcal{N}(d_1) - e^{-rT}K\mathcal{N}(d_1 - \sigma\sqrt{T})\right)\\ &= \mathcal{N}(d_1) + S\frac{\partial}{\partial S}\left(\mathcal{N}(d_1)\right) - \frac{\partial}{\partial S}\left(e^{-rT}K\mathcal{N}(d_1 - \sigma\sqrt{T})\right)\\ &= \mathcal{N}(d_1) + S\mathcal{N}'(d_1)\frac{\partial}{\partial S}(d_1) - e^{-rT}K\mathcal{N}'(d_1 - \sigma\sqrt{T})\frac{\partial}{\partial S}\left(d_1 - \sigma\sqrt{T}\right).\end{aligned}$$

Of course, $\mathcal{N}'$ is the density of a normalized Gaussian, i.e. $e^{-x^2/2}/\sqrt{2\pi}$, so we can calculate that

$$\begin{aligned}\mathcal{N}'\left(d_1 - \sigma\sqrt{T}\right) &= \frac{1}{\sqrt{2\pi}}\exp\left(\frac{-\left(d_1^2 - 2d_1\sigma\sqrt{T} + \sigma^2 T\right)}{2}\right) = \mathcal{N}'(d_1)\exp\left(d_1\sigma\sqrt{T} - \frac{\sigma^2 T}{2}\right)\\ &= \mathcal{N}'(d_1)\exp\left(\frac{\log(S/K) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}\sigma\sqrt{T} - \frac{\sigma^2 T}{2}\right)\\ &= \mathcal{N}'(d_1)\exp\left(\log(S/K)\right)\exp\left(\left(r + \frac{\sigma^2}{2}\right)T - \frac{\sigma^2 T}{2}\right) = \frac{Se^{rT}}{K}\mathcal{N}'(d_1)\end{aligned}$$

so that our entire expression for Δ simplifies to

$$\Delta = \mathcal{N}(d_1) + S\mathcal{N}'(d_1)\frac{\partial}{\partial S}(d_1) - e^{-rT}K\left(\frac{Se^{rT}}{K}\mathcal{N}'(d_1)\right)\frac{\partial}{\partial S}\left(d_1 - \sigma\sqrt{T}\right)$$

$$= \mathcal{N}(d_1) + S\mathcal{N}'(d_1)\left(\frac{\partial}{\partial S}(d_1) - \frac{\partial}{\partial S}(d_1) + \underbrace{\frac{\partial}{\partial S}\left(\sigma\sqrt{T}\right)}_{=0}\right) = \mathcal{N}(d_1). \tag{3.13a}$$

Calculations similar to the above give us values for the other greeks; in particular,

$$\Gamma = \frac{\mathcal{N}'(d_1)}{S\sigma\sqrt{T}}, \tag{3.13b}$$

$$\mathcal{V} = S\mathcal{N}'(d_1)\sqrt{T}. \tag{3.13c}$$

Many payoff functions exist beyond the vanilla call, which is why we have written our payoff as ϕ and avoided specification throughout this work. An example of a non–vanilla option is the binary option that pays \$1 if the price of the underlying at maturity exceeds the strike. That is, $\phi(S_T) = \mathbf{1}_{S_T>K}$ in the case of a binary European and $\phi(\bar{S}_T) = \mathbf{1}_{\bar{S}_T>K}$ for a binary Asian. It turns out that the value of a binary call option is identical to the Δ of a vanilla call option with identical parameters, i.e. $V_0^{\text{binary}} = \mathcal{N}(d_1)$ for European options as we saw in (3.13a). We can differentiate this equation to compute the greeks for a binary European call option in the same way we did above for the vanilla. Those computations give us

$$\Delta = \frac{e^{-rT}\mathcal{N}'(d_2)}{\sigma S_0\sqrt{T}}, \tag{3.14a}$$

$$\Gamma = -\frac{e^{-rT}d_1\mathcal{N}'(d_2)}{\sigma^2 S_0^2 T}, \tag{3.14b}$$

$$\mathcal{V} = -\frac{e^{-rT}\mathcal{N}'(d_2)d_1}{\sigma}. \tag{3.14c}$$

Testing

We are now ready to test the accuracy of our Malliavin estimators of Δ, Γ, and $\mathcal{V}$ against the analytic solutions in (3.13) and (3.14). We will use `MATLAB` to to simulate the evolution of S_t following a geometric Brownian motion a large number of times, compute the Malliavin formula for each simulation, and average the results to find the estimate of each Greek. In particular, we set the initial condition $S_0 = 100$, parameters $r = 0.05$, $\sigma = 0.10$, maturity $T = 1$, and time discretized into a number of small steps. We will generate 20000 sample paths. We issue the following commands, which invoke a custom script that can be found in appendix B:

```
>> S0 = 100; r = 0.05; sigma = 0.1; T = 1; NumPaths = 20000;
>> [S W] = GBMPaths(S0,r,sigma,T,NumPaths);
```

We use the generated paths to compute the Δ and $\mathcal{V}$ of a vanilla European call option with strike K using the Malliavin formulas (3.1) and (3.5) that we have developed. For $K = 105$, for example, we issue the following commands to `MATLAB`:

```
>> K = 105;
>> GBMEuroMalliavin('V',S,W,K,r,sigma,T);
```

To compute the analytic value for Δ, we use the built–in MATLAB implementation of the Black–Scholes formula expressed in (3.13a), i.e.

```
>> blsdelta(S0,K,r,T,sigma)
```

Similarly we use MATLAB's implementation of the Black–Scholes formula for Γ in (3.13b), i.e.

```
>> blsgamma(S0,K,r,T,sigma)
```

Of course, we could repeat the analysis for $\mathcal{V}$, but because of the proportionality $\Gamma = (S_0^2\sigma T)^{-1}\mathcal{V}$ that we saw earlier,[2] we know that the results will be identical (up to a scaling factor) to the ones in the Γ investigation.

We also perform these computations for a binary European call option, i.e. one with payoff $\phi(S_T) = \mathbf{1}_{S_T>0}$. These computations attempt to compare equations (3.1) for the Δ and (3.5) for the Γ with the analytic solutions (3.14a) and (3.14b) (respectively). For $K = 105$, for example, this is accomplished with the MATLAB commands

```
>> K = 105;
>> GBMEuroMalliavin('B',S,W,K,r,sigma,T);
```

Results

We can produce a plot to show how quickly our Malliavin estimates refine themselves towards the exact values as the number of simulations performed increases. This is purely qualitative, of course, but the results are telling. Figure 3.1 illustrates the quality of Malliavin estimator for the Δ of a vanilla European call option with strike $K = 100$. Note that this plot shows the result of half a million simulations, but also note the extremely compressed vertical axis—final error is .025516%. It is clear that the estimate approaches the correct value quickly, which confirms the idea that our estimator is unbiased.

We now attempt to quantitatively assess the accuracy of our Malliavin estimators. Having run our simulations for both binary and vanilla European call options, we have estimations of Δ and Γ for each payoff structure and for a variety of values for the strike K. We compare each approximation to the actual value obtained through the Black–Scholes formulas in (3.13a) and (3.14a) to compute the percent error. We are also interested in the variance of the estimator, which gives us a sense of how confident we can be in our results. When we speak of the variance of the estimator, we mean the variance of the results of the individual simulations $\{X_i\}$ where the Monte Carlo approximation is given by

$$\widehat{\mathbb{E}}\left[X\right] = \frac{1}{N}\sum_{i=1}^{N} X_i.$$

Because we want to compare estimators for different quantities throughout this work, we will normalize the variance by turning it into a signal–to–noise ratio with the equation

$$\text{SNR} = \frac{\widehat{\mathbb{E}}\left[X\right]}{\sqrt{\text{var}\left[\{X_i\}\right]}}.$$

[2]Note that this proportionality holds for the geometric Brownian motion model of price evolution but is not necessarily true for other models.

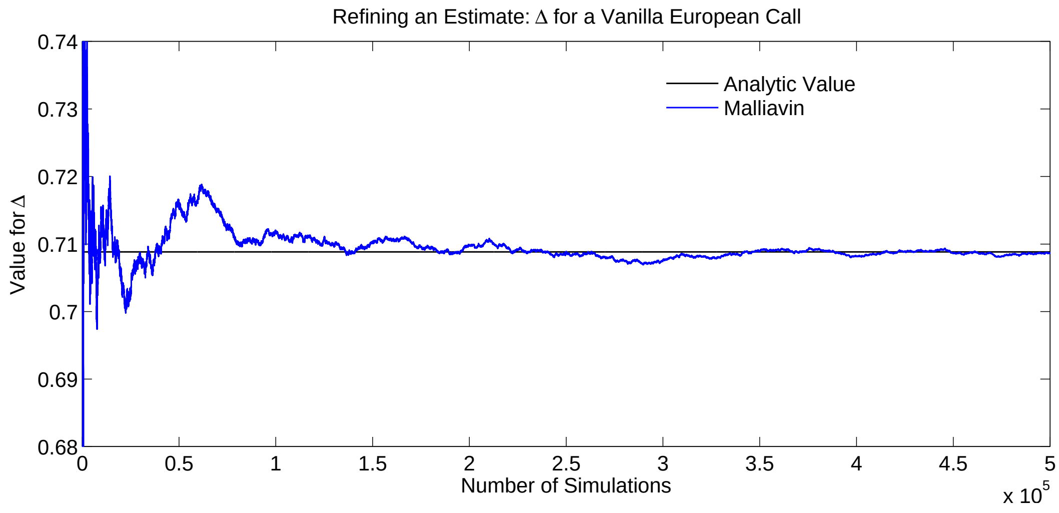


Figure 3.1: Monte Carlo estimation of the Δ of a vanilla European call option using Malliavin techniques. This plot confirms that our estimator is unbiased.

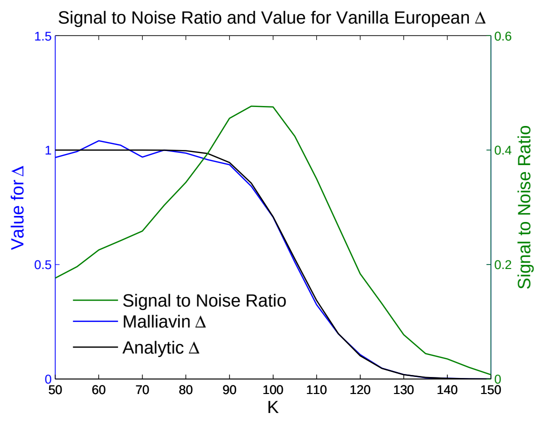
This equation makes sense for comparing estimators for various quantities because the standard deviation (square root of the variance) is in the same units of numéraire as the expectation. In general, a larger signal–to–noise ratio indicates a better estimator in the precise sense that the result given has smaller expected mean–square error.

We use the data from our tests to create plots comparing the actual value of the greek to our estimate across values of K. We add to the plot (on the right–hand vertical axis) the signal–to–noise ratio of the estimator. The four plots (one for each combination of payoff function and greek) are contained in figure 3.2. All of the results obtained in the above simulations are contained in Appendix A, starting on page 55. The results are grouped by payoff structure (vanilla or binary) and greek of interest (Δ or Γ) in tables A.1, A.2, A.3, and A.4.

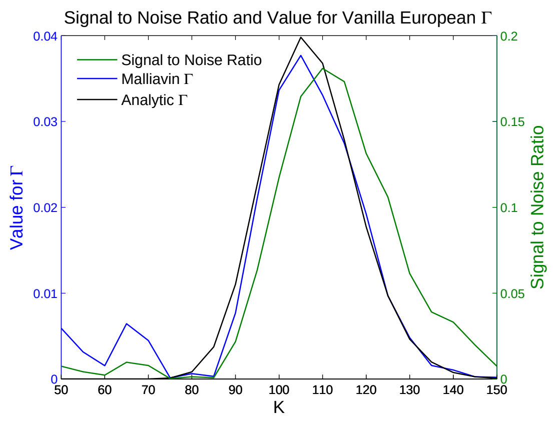
Our first observation is that the Malliavin approach quickly gives an accurate estimate of the actual value of the greeks in a reasonable time frame. A Sun V40z server with four AMD Opteron 848 2.2GHz CPUs was able to create the 20000 sample paths for the above computations in 0.322227 seconds; creating 40000 paths took 0.992807 seconds and cut error substantially across the board. The accuracy and usefulness of the Malliavin estimator is made graphically clear in figure 3.1, which shows rapid convergence to the analytic value of Δ for the case of $K = 100$ after approximately 3000 simulations.

The next observation is that the Malliavin estimator becomes increasingly less effective for values of K that are far away from the initial price S_0 (with the exception of the case of the Δ of a vanilla option, for which we retain excellent accuracy). More precisely, the technique becomes less effective the further away from "the money" the option is. Figure 3.3 makes this trend obvious by plotting the errors as a function of strike K. Note the log scale on the vertical axis: our errors seem to becoming astronomical as K becomes small.

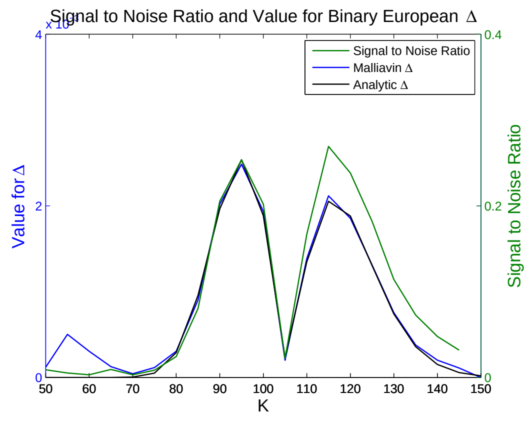
The large percent error is deceiving: the sensitivity of $\mathbb{E}[\phi]$ to any of its parameters becomes extremely small when a call option is so far into the money (i.e. when $S_0 >> K$), so Δ and Γ are almost zero. Though the percent errors are large, absolute error is small. Nonetheless, there



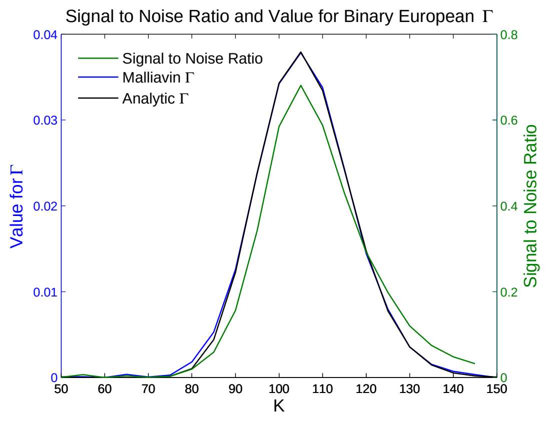
(a) Δ for a vanilla European call option



(b) Γ for a vanilla European call option



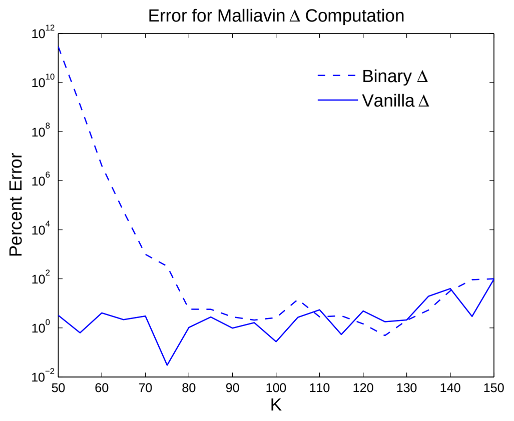
(c) Δ for a binary European call option



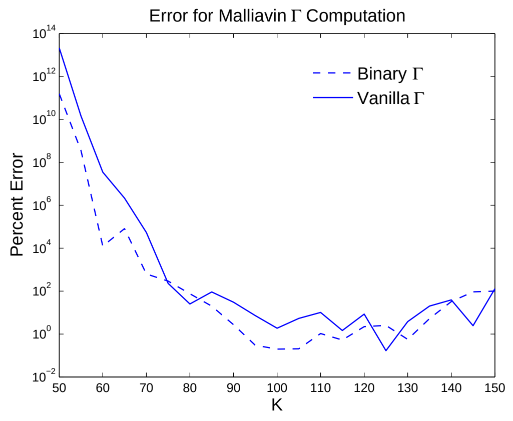
(d) Γ for a binary European call option

The data for these plots are contained in tables A.1 through A.4 beginning on page 55.

Figure 3.2: Plots showing actual values, estimates, and signal–to–noise ratio in the Malliavin estimates for the Δ and Γ of vanilla and binary European call options as a function of strike K.



(a) Vanilla



(b) Binary

The data for these plots are contained in tables A.1 through A.4 beginning on page 55.
Note log scale.

Figure 3.3: Plots showing the errors produced by Malliavin estimates for the Δ and Γ of European call options.

are errors, and they have a simple explanation: our estimator becomes inaccurate when the actual value of the greek gets extremely close to zero. For example, table A.2 shows that the Γ for a vanilla call option is on the order of 10^{-14} when $K = 50$ for $S_0 = 100$ (i.e. when the call is extremely far into the money). The Monte Carlo simulator lacks the resolution to distinguish among extremely small values, and so it returns a result that can be any orders of magnitude off. This is not a flaw in the Malliavin estimators, though, just a hurdle that is commonplace in numerical simulation of extremely small values.

Overall, we have strong evidence that our computations are correct and that the Malliavin calculus techniques we have described produce an unbiased estimator.

3.4.2 Comparison with Other Monte Carlo Estimators

In chapter 1, we saw a number of different approaches for computing sensitivities. In this section, we will compare two of them—the central finite difference method and the pathwise derivative method—with the Malliavin estimators that we constructed earlier in this chapter. For the reader's convenience, we briefly review the two methods against which we will be comparing our Malliavin estimators.

- The *central finite difference* estimator appears in approach 2 on page 5. To estimate $\phi'(\lambda_0) = \frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda_0\right]$, this approach has us estimate $\phi(\lambda_0 - \epsilon)$ and $\phi(\lambda_0 + \epsilon)$ with normal Monte Carlo methods and compute the approximation $\phi'(\lambda_0) \approx (\hat{\phi}(\lambda_0 + \epsilon) - \hat{\phi}(\lambda_0 - \epsilon))/(2\epsilon)$.
- The *pathwise derivative* estimator appears in approach 4 on page 8. To estimate $\phi'(\lambda_0) = \frac{\partial}{\partial\lambda}\mathbb{E}\left[X \mid \lambda_0\right]$, this approach has us compute $\frac{\partial}{\partial\lambda}X$ analytically, assuming such a computation is is possible near λ_0. If this is the case, then $\phi'(\lambda_0) = \mathbb{E}\left[\frac{\partial}{\partial\lambda}X \;\middle|\; \lambda_0\right]$, and we then then use normal Monte Carlo methods to compute $\widehat{\mathbb{E}}\left[\frac{\partial}{\partial\lambda}X \;\middle|\; \lambda_0\right]$, our estimate of $\phi'(\lambda_0)$.

Testing

To perform our analysis, we will—as in the previous section—be relying on custom `MATLAB` scripts. We will use the "master" scripts `MasterGBMEuro.m` and `MasterGBMAsian.m`, each of which calls a function to create Monte Carlo paths and then calls functions to use those paths to estimate Δ, Γ, and $\mathcal{V}$ with the Malliavin technique, the central finite difference technique, and the pathwise derivative technique. Again, all of the source code for scripts used in this analysis can be found in appendix B.

As an example, when we want to compute the Δ, Γ, and $\mathcal{V}$ of a vanilla European call option with We will be using parameters $S = 100, r = 0.05, \sigma = 0.1, T = 1$ to calibrate the geometric Brownian motion, and we will be performing $N = 40000$ simulations. A sample command (with $K = 100$) to analyze the data would be

```
>> K = 100; S0 = 100; r = 0.05; sigma = 0.1; T = 1; N = 40000;
>> MasterGBMEuro ('V',K,S0,r,sigma,T,N)
```

Results

At the end of these simulations, we have an incredible amount of data: for each strike K (we increment from 50 to 150 in steps of 5), we have three estimates (Malliavin, finite difference, and pathwise) for each of the three greeks $(\Delta, \Gamma, \mathcal{V})$ for both option types (European and Asian) and for both payoff structures (vanilla and binary)—over 700 different estimates in total. The following schematic indicates the location of the data; the tables of data themselves begin on page 57.[3]

	European			**Asian**		
	Δ	Γ	$\mathcal{V}$	Δ	Γ	$\mathcal{V}$
Vanilla	A.5	A.6	A.7	A.8	A.9	—
Binary	A.10	A.11	A.12	A.13	A.14	—

Before the in–depth analysis, we shall state the qualitative result of the simulations in the following observation.

Main Observation (Comparison of Monte Carlo Estimators of Sensitivities)**.** *The Malliavin estimators for sensitivities perform substantially better than the finite difference estimators and slightly worse than the pathwise derivative estimators. However, because the pathwise derivative method cannot be used in many problems, the Malliavin estimator is useful for practical computation.*

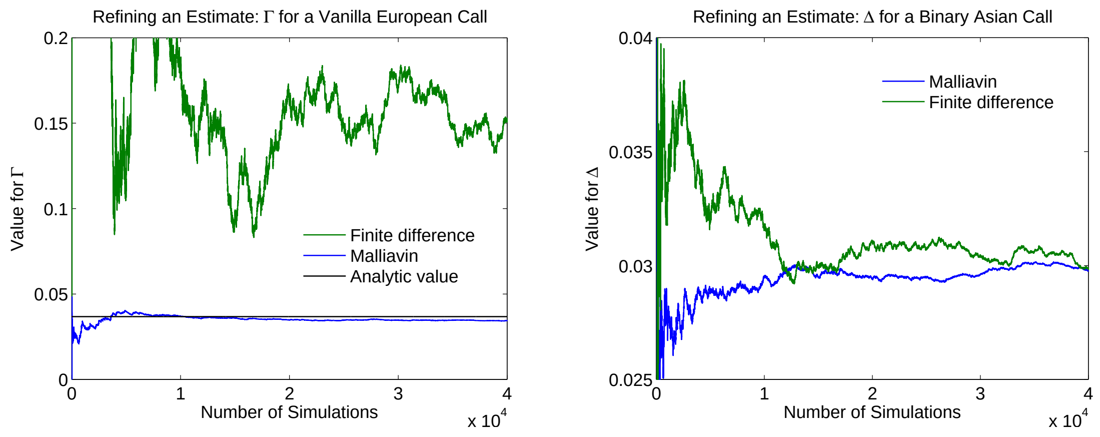
We will go about demonstrating this observation in three stages:

1. The Malliavin estimators perform better than the finite difference estimators by at least an order of magnitude.

2. The Malliavin estimator is better than the pathwise derivative estimator in that the latter cannot be used to compute two thirds of the sensitivities that we have considered in our study.

3. The pathwise derivative estimator cannot be used to compute two thirds of the sensitivities that we have considered in our study.

[3] Note that we do not have results in the right–most column (computations of $\mathcal{V}$ for Asian options) as we have not been able to derive or find in the literature a way to use Malliavin techniques to find the $\mathcal{V}$ of an Asian option.



(a) Γ for a vanilla European call, $K = 110$. (b) Δ for a binary Asian call, $K = 110$.

Figure 3.4: Refinement of Malliavin and finite difference estimators for different options.

Claim 1. The Malliavin estimators perform better than the finite difference estimators by at least an order of magnitude.

As we have discussed earlier in this section, we measure the quality of an estimator by both the error it produces and the signal–to–noise ratio. Our results indicate that the Malliavin estimator produces both smaller error and higher signal–to–noise ratios.

That the Malliavin estimator has lower error hardly surprises. It is an unbiased estimator, so we expect it to converge to the exact value in the long run. The finite difference estimator is biased, and we expect it to differ from the true value of the simulated greek in the limit. This theoretical difference between the two estimators is realized in the data, as is displayed in almost every table from A.5 to A.14. For the Δ of a vanilla European call with strike 95, for example, the Malliavin estimator produces an error of 0.05% while the finite difference estimator has an error of 4.52%. There is a general pattern is that the Malliavin estimator produces errors that are at least one magnitude smaller. Indeed, the real advantage of the Malliavin estimator comes in second order derivatives, which the finite difference method is terrible at computing: for a binary European call with strike 115, the Malliavin estimate differs from the analytic value of Γ by 1.90% while the finite difference method gives an error of 121.3%.

The two plots in figure 3.4 show the evolution of the Malliavin and finite difference estimates for a vanilla European's Γ (3.4a) and a binary Asian's Δ (3.4b), both with strike $K = 110$. In the left pane, we see excellent convergence on the analytic value by the Malliavin estimator of Γ and terrible performance of the finite difference estimator. This illustrates the advantage the Malliavin technique has in second order sensitivities. In the right pane, we do not know the analytic value for Δ of a binary Asian call (we have no closed form), but we do notice that both estimators approach a similar value—an apparent success of both models. However, the Malliavin estimator evolves much more smoothly, reaching a plateau relatively quickly and behaving less volatility. This motivates our next discussion of the Malliavin estimator's higher signal–to–noise ratio.

The signal–to–noise ratio is in this context the ratio

$$\frac{\widehat{\mathbb{E}}\left[X\right]}{\sqrt{\operatorname{var}\left[\{X_i\}\right]}}$$

where $\widehat{\mathbb{E}}[X] = \frac{1}{N}\sum X_i$. When we see an estimator evolution plot that moves violently and rapidly like the finite estimator lines in figure 3.4, we expect the estimator in question to have a low signal–to–noise ratio. In general, the results of our simulations indicate that the Malliavin estimator has a better signal–to–noise ratio than the finite difference estimator across the board. Figure 3.5 shows signal–to–noise ratio as a function of strike K for four different greeks from our simulation pool (these four graphs represent about 40% of the data available). All plots indicate better performance from the Malliavin estimator than from the finite difference one.

Figure 3.5 shows other interesting behavior, in particular a difference between the Malliavin estimator and the finite difference estimator for computing Δ of vanilla options as K becomes small. The finite difference estimator's signal–to–noise ratio reaches a plateau and seems to stay there while the Malliavin estimator sees substantial falloff in figures 3.5a and 3.5d. We can explain the stability of the finite difference estimator for low K as follows: for small K, the option is extremely likely to finish in the money, and movement in the current price will accordingly flow through to the payoff function in (almost) direct proportion. Thus $\mathbb{E}[\phi(\cdot)]$ is almost linear as a function of current price S for low K. In this case, the linear (secant) approximation that the finite difference method uses of ϕ is well justified, and its signal is consistent as K falls.

Claim 2. The pathwise derivative estimator, when it can be used, slightly outperforms the Malliavin estimator.

Both the pathwise derivative and the Malliavin estimators are unbiased, and we expect neither to show error in the long run. Nonetheless, both methods do produce some error after our approximation with $N = 40000$ simulations. We compare these two methods with respect to both the errors produced (in relation to an analytic solution if one exists) and the signal–to–noise ratios.

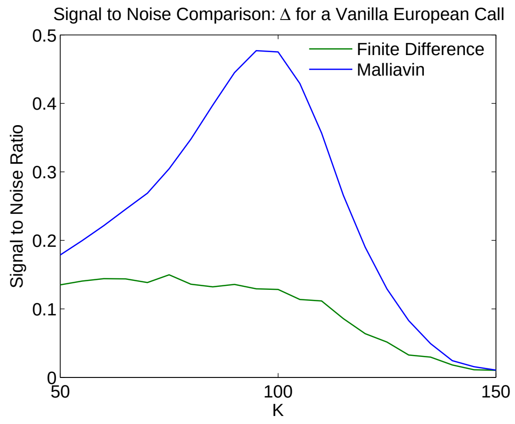
In figure 3.6, we show two plots comparing the errors of the Malliavin and pathwise derivative estimators for a vanilla European call. As a general rule, the two errors are very similar, though the pathwise derivative method appears to have a slight advantage in offering useful results. That the plots appear to indicate errors growing extremely large is misleading: tables A.5 and A.7 show that the large percent errors are indeed tiny errors on extremely small numbers.

We next compare the signal–to–noise ratios of the Malliavin and pathwise derivative estimators. Figure 3.7 compares these ratios for the two estimators for the Δ ad $\mathcal{V}$ of a vanilla European call option. These plots show a clear distinction between the two estimators—the pathwise derivative estimator has a substantially better signal–to–noise ratio in both cases. Thus, all things being equal, we could have more confidence in a sensitivity estimate given by the pathwise derivative method than by the Malliavin method.

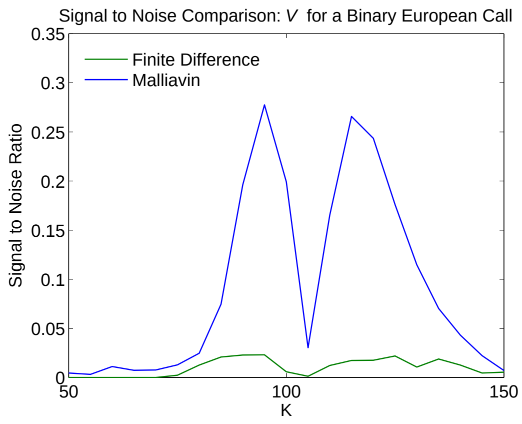
There is not a consistent explanation for the pathwise derivative estimator slightly outperforming the Malliavin estimator, and the difference between the two is not significant.

Claim 3. The Malliavin estimator is better than the pathwise derivative estimator in that the latter cannot be used to compute two thirds of the sensitivities that we have considered in our study.

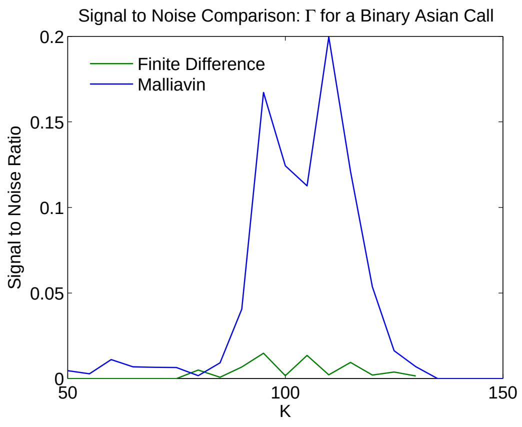
When we discussed the pathwise derivative estimator in chapter 1, we discovered its failure to estimate $\frac{\partial}{\partial\lambda}\mathbb{E}[f(X)]$ whenever f has a jump discontinuity (proposition 1.2.1, proposition 1.2.2). In the application of estimating greeks, this failure is devastating. A binary option has payoff $\phi(\cdot) = \mathbf{1}_{\cdot > K}$, exactly the situation discussed in proposition 1.2.1, and so Monte Carlo estimation fails to compute its price sensitivity to any parameter. A vanilla option does not suffer so—its



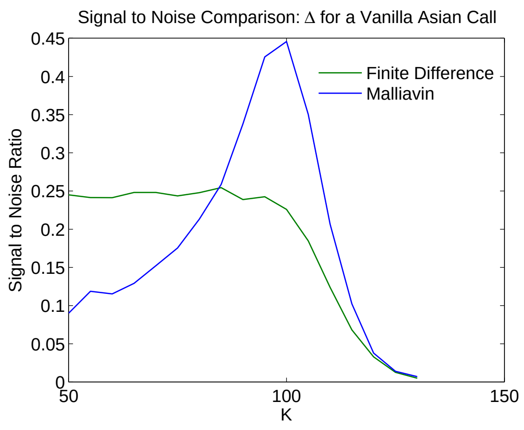
(a) Δ for a vanilla European call.



(b) $\mathcal{V}$ for a binary European call.

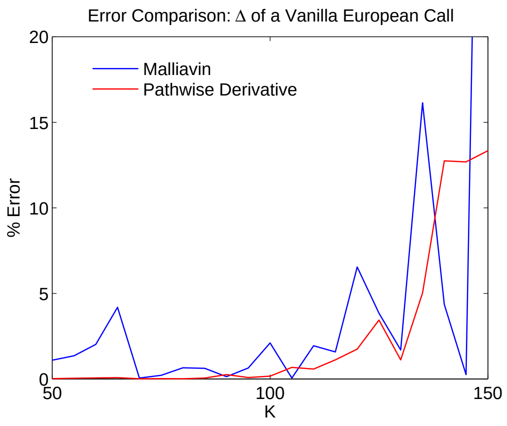


(c) Γ for a binary Asian call.

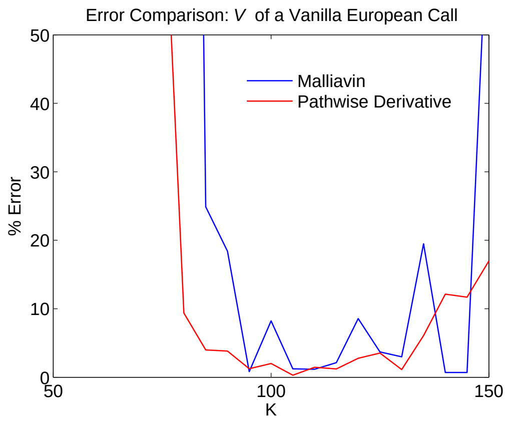


(d) Δ for a vanilla Asian call.

Figure 3.5: Comparison of signal–to–noise ratios for Malliavin and finite difference estimators.

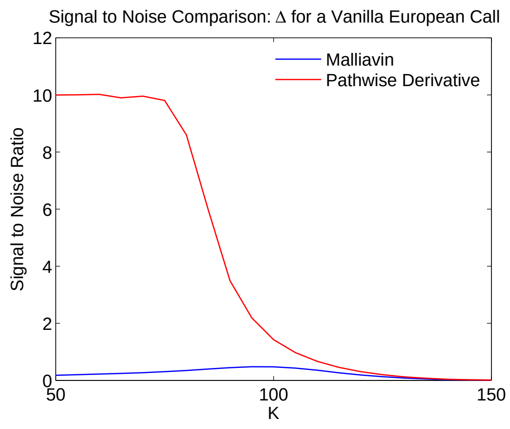


(a) Δ for a vanilla European call.

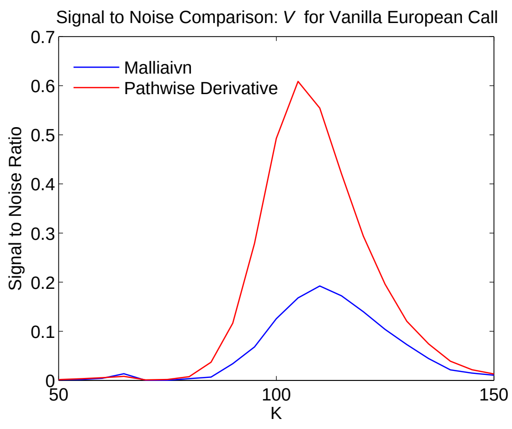


(b) $\mathcal{V}$ for a vanilla European call.

Figure 3.6: Comparison of the percent error for Malliavin and pathwise derivative estimators.



(a) Δ for a vanilla European call.



(b) $\mathcal{V}$ for a vanilla European call.

Figure 3.7: Comparison of the signal–to–noise ratios for Malliavin and pathwise derivative estimators.

payoff $\phi(\cdot) = (\cdot - K)^+$ is continuous—but its first derivative $\phi'(\cdot) = \mathbf{1}_{\cdot > K}$ does. It follows that the pathwise derivative estimator can be used to approximate $\Delta = \frac{\partial}{\partial S_0}\mathbb{E}[\phi(\cdot)]$ and $\mathcal{V} = \frac{\partial}{\partial \sigma}\mathbb{E}[\phi(\cdot)]$, but it *cannot* be used to estimate a second derivative like $\Gamma = \frac{\partial^2}{\partial S_0^2}\mathbb{E}[\phi(\cdot)]$. The following diagram summarizes the failures of the pathwise derivative estimator:

European

	Δ	Γ	$\mathcal{V}$
Vanilla	$\surd$	$\times$	$\surd$
Binary	$\times$	$\times$	$\times$

Asian

	Δ	Γ	$\mathcal{V}$
Vanilla	$\surd$	$\times$	$\surd$
Binary	$\times$	$\times$	$\times$

Applicability of the pathwise derivative estimator.

The beauty of the Malliavin estimators is that they are payoff–agnostic: once we have derived the Malliavin estimator for, say, the Γ of an Asian option (3.10), that estimator will work regardless of the payoff function $\phi(\cdot)$ we choose. All of our Malliavin estimator formulas took the form

$$\frac{\partial}{\partial \lambda}\mathbb{E}[f(X)] = \mathbb{E}[f(X)H];$$

once we determine H for a certain greek, solving a problem is just a matter of plugging the specified payoff into our formula.

3.5 Extensions

In this final section, we investigate two extensions of the ideas we have presented in this work. The first is the derivation of Malliavin estimators for greeks when prices do not follow a geometric Brownian motion. In particular, we will find an equation for the Δ of a European call option when a stochastic volatility model is used.

The second extension we present is an application of the Malliavin calculus to the Monte Carlo approximation of conditional expectations. This is on the surface a very different problem than the computation of sensitivities, but the two end up sharing some key characteristics that make them well–suited to the use of Malliavin calculus to form an estimator. Once we have found the Malliavin estimator for a problem with conditional expectations, we will work out an example to demonstrate how one could perform the computations involved.

3.5.1 Other Pricing Models

Throughout this chapter, we have assumed that prices evolve according to a geometric Brownian motion. This assumption, which Black and Scholes also made [4] when deriving their model, is defined by the stochastic differential equation

$$\mathrm{d}S_t = rS_t \ \mathrm{d}t + \sigma S_t \ \mathrm{d}W_t$$

and yields the price process

$$S_t = S_0 \exp((r - \sigma^2/2)t + \sigma W_t).$$

We have seen that this assumption leads to facile computations of Malliavin weights in the preceding sections. Facility is not plausibility, though, and we noted in section 3.1 some problems with the model.

Financial mathematicians have developed other models for prices that better reflect market behavior. In very general terms, we can describe a continuous–time models with the Itô drift–diffusion process[4]

$$\mathrm{d}S_t = r(t, S_t) \ \mathrm{d}t + \sigma(t, S_t) \ \mathrm{d}W_t, \tag{3.15}$$

and specifying a model is a matter of giving definitions to the interest rate and volatility processes r and σ. In the Black–Scholes model, for example, $r(t, S_t) = rS_t$ and $\sigma(t, S_t) = \sigma S_t$.

Generalized Malliavin Estimators

Suppose we wanted to compute Δ for a European option when the price S_t follows some diffusion with general form (3.15). If we recall our integration–by–parts rule in proposition 2.3.1, we have

$$\Delta = \mathbb{E}\left[f'(X)Y\right] = \mathbb{E}\left[f(X)\delta\left(u\right)\right]$$

where u was a process that we could specify. As Benhamou points out, we will have $X = S_t$ and $Y = \frac{\partial}{\partial S_0} S_t$ [2]. We now try to find u so that we can perform a Monte Carlo simulation.

The duality principle of Malliavin calculus (2.3) tells us that

$$\begin{aligned}\Delta = \mathbb{E}\left[\phi(S_t)\delta(u)\right] &= \mathbb{E}\left[\int_0^T (D_t\phi(S_t))u_t \ \mathrm{d}t\right] \\ &= \mathbb{E}\left[\int_0^T \phi'(S_t)(D_tS_t)u_t \ \mathrm{d}t\right] = \mathbb{E}\left[\phi'(S_t)\int_0^T (D_tS_t)u_t \ \mathrm{d}t\right]\end{aligned}$$

We now state without proof a result from the Malliavin calculus: if S_t follows an Itô drift–diffusion model like (3.15), then the Malliavin derivative D_sS_t is given by

$$D_sS_t = \sigma(s, S_s)Y_tY_s^{-1}\mathbf{1}_{s<t}$$

[4]Many continuous models use Itô drift–diffusion processes, but not all must.

where $Y_t = \frac{\partial}{\partial S_0} S_t$, which is sometimes referred to as the *first variation process.* A complete proof of this equality is given in [20, ch. 1], and its machinery is beyond the scope of this work.

Resuming our derivation of a formula for Δ, we have

$$\Delta = \mathbb{E}\left[\phi'(S_t) \int_0^T \left(\frac{\sigma(s, S_s)Y_t}{Y_s}\right) u_t \text{ d}t\right],$$

, but then, we also have $\Delta = \mathbb{E}\left[\phi'(S_t)Y\right]$. The two expressions for Δ must be equivalent, so we obtain

$$\mathbb{E}\left[\phi'(S_t)Y_t\right] = \mathbb{E}\left[\phi'(S_t) \int_0^T \left(\frac{\sigma(s, S_s)Y_t}{Y_s}\right) u_t \text{ d}t\right], \tag{3.16}$$

and our task is finding u such that (3.16) holds. We can choose, as in [2]

$$u_t = \frac{Y_s}{\sigma(s, S_s)T}$$

such that the integral on the right hand side of (3.16) is

$$\int_0^T \left(\frac{\sigma(s, S_s)Y_t}{Y_s}\right) \frac{Y_s}{\sigma(s, S_s)T} \text{ d}t = \frac{1}{T}\int_0^T Y_t \text{ d}t = Y_t,$$

which satisfies (3.16). We obtain a general Malliavin estimator for the Δ of a European call option by plugging this u into the integration–by–parts rule at the beginning of this section, giving us

$$\Delta = \mathbb{E}\left[\phi(S_t)\frac{1}{T}\delta\left(\frac{Y_s}{\sigma(s, S_s)}\right)\right] \tag{3.17}$$

This requires, of course, that $\sigma(s, S_s) \neq 0$ (except maybe on a set of measure zero). In principle, that requirement always satisfied as any reasonable financial model is non–deterministic. At this point, we are unable to further compute this estimator without specifying a model for S_t.

A Malliavin Estimator for a Stochastic Volatility Model

A model due to Heston [13] adds a stochastic volatility to the basic geometric Brownian framework. Using the general framework of (3.15), this model has $r(t, S_t) = rS_t$ as in Black–Scholes, but the volatility is given by the function $\sigma(t, S_t) = \sqrt{\nu_t}S_t$ where ν_t is itself a process defined by a second stochastic differential equation. The complete description of the Heston model is

$$\begin{aligned} \text{d}S_t &= rS_t \text{ d}t + \sqrt{\nu_t}S_t \text{ d}W_t^{(1)} \\ \text{d}\nu_t &= \kappa(\theta - \nu_t) \text{ d}t + \xi\sqrt{\nu_t} \text{ d}W_t^{(2)}. \end{aligned}$$

Here θ is a theoretical "long term" volatility that ν is drawn towards, κ controls how strongly ν is drawn towards θ, ξ is extent to which ν is by randomness (second–order volatility), and $W_t^{(2)}$ is a Wiener process that is distinct from—though could be specified to correlate with—the first Wiener process $W_t^{(1)}$.

We now apply the general formula (3.17) for the Δ of a European call option to the Heston model. This computation, as performed in [2], is easy at this point. For the Heston model, the first variation process Y_t is given by

$$Y_t = \frac{\partial}{\partial S_0} S_t = \frac{S_t}{S_0},$$

which we saw to be true for the geometric Brownian motion as well. We thus have

$$\begin{aligned}
\Delta &= \mathbb{E}\left[\phi(S_t)\frac{1}{T}\delta\left(\frac{Y_s}{\sigma(s, S_s)}\right)\right] \\
&= \mathbb{E}\left[\phi(S_t)\frac{1}{T}\delta\left(\frac{S_s}{S_0\sqrt{\nu_s}S_s}\right)\right] = \mathbb{E}\left[\phi(S_t)\frac{1}{S_0 T}\delta\left(\frac{1}{\sqrt{\nu_s}}\right)\right] \\
&= \mathbb{E}\left[\phi(S_t)\frac{1}{S_0 T}\int_0^T \frac{\mathrm{d}W_t^{(1)}}{\sqrt{\nu_s}}\right]
\end{aligned}$$

where the conversion from the Skorohod to the Itô integral is justified because $\frac{1}{\sqrt{\nu_s}}$ is adapted. This final expression is the Malliavin estimator for the Δ of a European call option with payoff ϕ.

3.5.2 Conditional Expectations and Malliavin Calculus

In this section, we will apply the ideas we have developed about computing sensitivities to a seemingly distinct problem that shares some crucial features in common with the computation of greeks.

Many problems in applied probability theory demand the calculation of conditional expectation. For example, a biologist interested in the simulating a population could construct a model that required assumptions about the lifespans of the individual creatures. The researcher could make a simplifying assumption like "suppose each of the members of the first generation lives for n days" that makes the computation more feasible and run the simulation a number of times to calculate an expectation of the size P_T of the population at time T conditioned on the fact that the lifespan L_1 of the first generation of organisms was n days, or $\mathbb{E}[P_T \mid L_1 = n]$.

Sometimes, conditioning has an opposite effect, making a problem more difficult to solve. In the context of our financial modelling sections, one could ask for an expectation of some function $\phi(S_T)$ of a stock price on day T conditioned on the stock price being in some specified range $[a, b]$ on day $T/2$. Formally, we would be trying to estimate

$$\mathbb{E}\left[\phi(S_T) \;\middle|\; S_{T/2} \in [a, b]\right].$$

Such a what–if question could be important for a practitioner performing risk–controls by investigating a portfolio's performance in various economic climates.

Supposing that we had to perform this computation with Monte Carlo simulation, our methods become extremely inefficient: when we generate sample paths for the stock price evolution, we are forced to throw out all paths that do not have $S_{T/2} \in [a, b]$. This decreases the accuracy of the result substantially, and, if $[a, b]$ is small enough, could make Monte Carlo simulation completely useless.

We will use a relative of our Malliavin integration–by–parts formula to correct this deficiency. Heuristically, we can think of the requirement that $S_{T/2}$ falls in $[a, b]$ as a step function being applied to $\phi(S_T)$ that makes our computed value zero whenever $S_{T/2}$ falls outside a certain value.

By using the Malliavin calculus to rewrite $\mathbb{E}\left[\phi(S_T) \mid S_{T/2} \in [a,b]\right]$ with the anti–derivative of that step function, we will obtain an estimator that many of the Monte Carlo paths hit.

Here we follow the outline sketched by [8] for a way to use Malliavin calculus techniques to simplify the computation of conditional expectations. Instead of conditioning on $S_{T/2}$ being in a certain range, we will make the very general computation of $\mathbb{E}[\phi(F) \mid G = 0]$.

Proposition 3.5.1. *Suppose we have smooth random variables F and G in $\mathcal{S}$, i.e. F and G admit Malliavin derivatives D_tF and D_tG. Furthermore, suppose that there exists some process $u = \{u_t\}$ such that*

$$\mathbb{E}\left[\int_0^T (D_tG)u_t \,\mathrm{d}t\right] = 1.$$

Then

$$\mathbb{E}[\phi(F) \mid G = 0] = \frac{\mathbb{E}\left[\phi(F)\mathbf{1}_{G>0}\delta(u) - \phi'(F)\mathbf{1}_{G>0}\int_0^T (D_tF)u_t \,\mathrm{d}t\right]}{\mathbb{E}[\mathbf{1}_{G>0}\delta(u)]} \tag{3.18}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a well-behaved function and the D and $\delta(\cdot)$ operators signify the Malliavin derivative and Skorohod integral as in earlier sections.

Proof. Conditional expectation is defined with Bayes's Law, which states that [22]

$$\mathbb{E}[\phi(F)|G = 0] = \frac{\mathbb{E}[\phi(F)\delta_0(G)]}{\mathbb{E}[\delta_0(G)]}, \tag{3.19}$$

where $\delta_0(\cdot)$ is the Dirac δ function that we have seen before. We interpret this equation as an instruction to only give $\phi(F)$ any weight when $G = 0$ exactly with a normalizing factor. This quantity is impossible to compute with a Monte Carlo simulator because for a random variable G with support of positive measure, $G \neq 0$ almost surely. This is a very similar problem to the one we encountered when trying to apply the path–wise method to compute the Γ of an option: almost all paths miss the Dirac delta function of a random variable.

We will prove the assertion of the theorem by showing that the numerator and denominator of (3.19) are equal to the numerator and denominator (respectively) of (3.18) almost surely. Both proofs will rely on the duality principle of Malliavin calculus (2.3), which we recall here for convenience:

$$\mathbb{E}[F\delta(u)] = \mathbb{E}\left[\int_0^T (D_tF)u_t \,\mathrm{d}t\right].$$

We stress at this point that this derivation only holds when F and G are in the space $\mathcal{S}$ of random variables that have Malliavin derivatives. We also note the notational conflict that $\delta(\cdot)$ represents the Skorohod integral while $\delta_0(x)$ represents the Dirac δ function; the two should not be confused.

We assume that some process exists such that $\int_0^T (D_tG)u_t \,\mathrm{d}t = 1$. We multiply the denominator of (3.19) by this integral, which under our assumption does not affect its value. That is,

$$\mathbb{E}[\delta_0(G)] = \mathbb{E}\left[\delta_0(G)\underbrace{\int_0^T (D_tG)u_t \,\mathrm{d}t}_{=1}\right] = \mathbb{E}\left[\int_0^T \delta_0(G)(D_tG)u_t \,\mathrm{d}t\right].$$

Now, because $\delta_0(G) = \frac{\partial}{\partial x}\mathbf{1}_{G>0}$, which we saw on page 9 in the proof of proposition 1.2.1, the chain rule for the Malliavin derivative tells us that

$$D_t(\mathbf{1}_{G>0}) = \frac{\partial}{\partial x}(\mathbf{1}_{G>0})D_tG = \delta_0(G)D_tG,$$

which we recognize in the integrand above. Substitution with this transformation yields

$$\mathbb{E}\left[\delta_0(G)\right] = \mathbb{E}\left[\int_0^T D_t\left(\mathbf{1}_{G>0}\right)u_t \,\mathrm{d}t\right],$$

and an application of the duality principle gives us

$$\mathbb{E}\left[\delta_0(G)\right] = \mathbb{E}\left[\mathbf{1}_{G>0}\delta(u)\right],$$

which completes the proof that the denominators of (3.19) and (3.18) are identical.

As for the almost sure equality of the numerators, we begin by applying the product rule for the Malliavin derivative to compute $D_t(\phi(F)\mathbf{1}_{x>0}(G))$, i.e.

$$\begin{aligned} D_t(\phi(F)\mathbf{1}_{G>0}) &= \phi(F)D_t(\mathbf{1}_{G>0}) + \mathbf{1}_{G>0}D_t(\phi(F)) \\ &= \phi(F)\delta_0(G)D_tG + \mathbf{1}_{G>0}\phi'(F)D_tF \end{aligned}$$

We next multiply both sides of this equality by u_t and integrate with respect to t over the time domain:

$$\int_0^T D_t(\phi(F)\mathbf{1}_{G>0})u_t \,\mathrm{d}t = \int_0^T \phi(F)\delta_0(G)(D_tG)u_t \,\mathrm{d}t + \int_0^T \mathbf{1}_{G>0}\phi'(F)(D_tF)u_t \,\mathrm{d}t.$$

Taking expectations,

$$\mathbb{E}\left[\int_0^T D_t(\phi(F)\mathbf{1}_{G>0})u_t \,\mathrm{d}t\right] = \mathbb{E}\left[\phi(F)\delta_0(G)\underbrace{\int_0^T (D_tG)u_t \,\mathrm{d}t}_{=1}\right] + \mathbb{E}\left[\mathbf{1}_{G>0}\phi'(F)\int_0^T (D_tF)u_t \,\mathrm{d}t\right].$$

Rearranging terms,

$$\mathbb{E}\left[\int_0^T D_t(\phi(F)\mathbf{1}_{G>0})u_t \,\mathrm{d}t\right] - \mathbb{E}\left[\mathbf{1}_{G>0}\phi'(F)\int_0^T (D_tF)u_t \,\mathrm{d}t\right] = \mathbb{E}\left[\phi(F)\delta_0(G)\right].$$

With this manipulation, we recognize on the right hand side the numerator of (3.19). As for the left hand side, we use the duality principle to transform the leftmost expectation as

$$\mathbb{E}\left[\int_0^T D_t(\phi(F)\mathbf{1}_{G>0})u_t \,\mathrm{d}t\right] = \mathbb{E}\left[\phi(F)\mathbf{1}_{G>0}\delta(u)\right],$$

which completes the proof that the numerators of (3.19) and (3.18) are identical. We have thus shown

$$\frac{\mathbb{E}\left[\phi(F)\delta_0(G)\right]}{\mathbb{E}\left[\delta_0(G)\right]} = \frac{\mathbb{E}\left[\phi(F)\mathbf{1}_{G>0}\delta(u) - \phi'(F)\mathbf{1}_{G>0}\int_0^T (D_tF)u_t \,\mathrm{d}t\right]}{\mathbb{E}\left[\mathbf{1}_{G>0}\delta(u)\right]},$$

which completes the proof. □

The right hand side of equation (3.18) is the Malliavin estimator for the conditional expectation $\mathbb{E}[\phi(F) \mid G = 0]$. Once we have chosen a process u such that $\mathbb{E}\left[\int_0^T (D_tG)u_t \, \mathrm{d}t\right] = 1$, we compute the numerator and denominator separately using Monte Carlo methods.

Remark 3.5.2. Necessary to our argument was the existence of a process $u = \{u_t\}$ with the property that

$$\mathbb{E}\left[\int_0^T (D_tG)u_t \, \mathrm{d}t\right] = 1.$$

It is worth noting that this requirement is not particularly stringent: if $D_tG \neq 0$ almost everywhere, we can let

$$u_t = \begin{cases} \frac{1}{TD_tG} & D_tG \neq 0 \\ 1 & D_tG = 0 \end{cases}$$

and we have constructed a process u that satisfies the requirement. Of course, having $D_tG \neq 0$ almost everywhere is sufficient but not necessary for the existence of such a process. Moreover, even if $D_tG \neq 0$ *a.e*, other suitable processes might exist, and we have some degree of latitude to choose one that can make our calculation simpler. Recalling the numerator of (3.18), namely

$$\mathbb{E}\left[\phi(F)\mathbf{1}_{G>0}\delta(u) - \phi'(F)\mathbf{1}_{G>0}\int_0^T (D_tF)u_t \, \mathrm{d}t\right],$$

if we are able to choose a process u such that $\int_0^T (D_tF)u_t \, \mathrm{d}t = 0$, then we greatly simplify our calculation.

Example 3.5.3. We can demonstrate analytically that our formula (3.18) is correct in the simple case where $F = G = W_T$ and $\phi(x) = x$; we clearly have $F, G \in \mathcal{S}$, and ϕ meets the necessary regularity conditions. Recalling the assumptions of Proposition 3.5.1, we need to find a process u such that $\mathbb{E}\left[\int_0^T (D_tG)u_t \, \mathrm{d}t\right] = 1$. Since $G = W_T$, we have already calculated that $D_tG = \mathbf{1}_{t<T}$ in (2.2); since $t < T$ on our entire domain, we can simplify this to $D_tG = 1$. Then, by setting u identically equal to $\frac{1}{T}$, $\mathbb{E}\left[\int_0^T (D_tG)u_t \, \mathrm{d}t\right] = \mathbb{E}\left[\int_0^T \frac{1}{T} \, \mathrm{d}t\right] = 1$. Our formula (3.18) for conditional expectation then asserts that

$$\begin{aligned} \mathbb{E}[\phi(F)|G = 0] &= \frac{\mathbb{E}\left[\phi(F)\mathbf{1}_{G>0}\delta(u) - \phi'(F)\mathbf{1}_{G>0}\int_0^T (D_tF)u_t \, \mathrm{d}t\right]}{\mathbb{E}[\mathbf{1}_{G>0}\delta(u)]} \\ &= \frac{\mathbb{E}\left[\phi(W_T)\mathbf{1}_{W_T>0}\delta(\frac{1}{T}) - \phi'(W_T)\mathbf{1}_{W_T>0}\int_0^T (D_tW_T)\frac{1}{T} \, \mathrm{d}t\right]}{\mathbb{E}\left[\mathbf{1}_{W_T>0}\delta(\frac{1}{T})\right]} \end{aligned} \tag{3.20}$$

We use the facts that $\phi'(x) = 1$ and $D_tW_T = \mathbf{1}_{t<T} \equiv 1$ to rewrite the numerator of (3.20):

$$\begin{aligned} \mathbb{E}\left[\phi(W_T)\mathbf{1}_{W_T>0}\delta\left(\frac{1}{T}\right) - \phi'(W_T)\mathbf{1}_{W_T>0}\int_0^T (D_tW_T)\frac{1}{T} \, \mathrm{d}t\right] &= \mathbb{E}\left[\mathbf{1}_{W_T>0}\left(W_T\frac{\delta(1)}{T} - \int_0^T \frac{1}{T} \, \mathrm{d}t\right)\right] \\ &= \mathbb{E}\left[\mathbf{1}_{W_T>0}\frac{W_T^2}{T}\right] - \mathbb{E}[\mathbf{1}_{W_T>0}]. \end{aligned}$$

Note that we have used our computation (2.4) that $\delta(1) = W_T$. Now, we know that $W_T \sim \mathcal{N}(0, T)$, so $\mathbb{P}\{W_T > 0\} = \frac{1}{2}$. Thus $\mathbb{E}[\mathbf{1}_{W_T>0}] = \frac{1}{2}$. As for the left hand expectation, we have

$$\begin{aligned}\mathbb{E}\left[\mathbf{1}_{W_T>0}\frac{W_T^2}{T}\right] &= \frac{1}{T}\int_0^\infty x^2\left(\frac{e^{-x^2}/2T}{\sqrt{2\pi T}}\right)\,\mathrm{d}x = \frac{1}{T}\int_0^{-\infty}(-x)^2\left(\frac{e^{-(-x)^2}/2T}{\sqrt{2\pi T}}\right)\,(-\mathrm{d}x)\\ &= \frac{1}{T}\int_{-\infty}^0 x^2\left(\frac{e^{-x^2}/2T}{\sqrt{2\pi T}}\right)\,\mathrm{d}x = \mathbb{E}\left[\mathbf{1}_{W_T<0}\frac{W_T^2}{T}\right].\end{aligned}$$

That is to say that W_T^2 has the same distribution for $W_T > 0$ as it does for $W_T < 0$. Of course, $\mathbb{E}[\mathbf{1}_{X>0}] + \mathbb{E}[\mathbf{1}_{X<0}] = \mathbb{E}[X]$ for any random quantity X, or in this case,

$$\begin{aligned}\mathbb{E}\left[\mathbf{1}_{W_T>0}\frac{W_T^2}{T}\right] + \mathbb{E}\left[\mathbf{1}_{W_T<0}\frac{W_T^2}{T}\right] &= \mathbb{E}\left[\frac{W_T^2}{T}\left(\mathbf{1}_{W_T>0} + \mathbf{1}_{W_T<0}\right)\right]\\ &= \frac{1}{T}\mathbb{E}\left[W_T^2\right] = \frac{1}{T}\left(\underbrace{\mathrm{var}\left[W_T\right]}_{=T} + \underbrace{\mathbb{E}\left[W_T\right]^2}_{=0}\right) = 1,\end{aligned}$$

so the fact that $\mathbb{E}\left[\mathbf{1}_{W_T>0}\frac{W_T^2}{T}\right] = \mathbb{E}\left[\mathbf{1}_{W_T<0}\frac{W_T^2}{T}\right]$ implies that $\mathbb{E}\left[\mathbf{1}_{W_T>0}\frac{W_T^2}{T}\right] = \frac{1}{2}$. Thus the numerator of (3.20) is $\frac{1}{2} - \frac{1}{2} = 0$, and we conclude that

$$\mathbb{E}\left[W_T | W_T = 0\right] = 0.$$

Of course, this result is trivial, but it is a confirmation that our equation (3.18) describes an unbiased estimator for the conditional expectation $\mathbb{E}\left[\phi(F) \mid G = 0\right]$.

Conclusion

We saw both in our discussion of sensitivities and in our closing discussion of an application to computing conditional expectations, that estimating a Dirac δ function of a random variable is hard with numerical methods. This difficulty makes integration–by–parts rules like those that we developed in chapter 2 particularly useful. In the same way that the integration–by–parts rule of the standard integral calculus helps us deal with functions that are particularly difficult to integrate, the rule we found in proposition 2.3.1 lets us rewrite problems in ways that are possibly easier to solve. These Malliavin integration–by–parts rules have the added benefit that their use does not require knowledge of the distributions of the random variables involved. We have seen that these estimators allow us to formulate expressions for sensitivities of financial quantities that would have been difficulty to compute with other methods. In particular, the formula we found for the Γ of an Asian call option, which does not appear in the literature, is particularly useful because we have no analytic expression for the same quantity and because other estimators for second derivatives like Γ are not accurate or efficient.

The Malliavin estimators are not hands–down winners for computing sensitivities. We saw that they did a somewhat worse job than the path–wise derivative estimator when the latter could be used. Instead of being excellent all–purpose tools, these estimators seem to be most useful when we encounter random variables—in the context of our financial application, payoff functions—that are pathological in their discontinuity. The Malliavin estimators are bad when the going is good and good when the going is bad, and that makes them a good addition to a practitioner's tool–kit.

APPENDIX A

Numerical Results

In section 3.4, we investigated the performance of our Malliavin formulas by comparing them with analytic approaches (where possible) and the other numerical approaches for computing sensitivities. The following pages contains all of the data from our simulations in table form. All plots in the text showing the results of simulation are graphical representations of the data in this section and are appropriately cross–referenced as such.

K	**Mall. Δ**	**B.–S. Δ**	**Error**	**SNR**
50	0.9677	1	3.23%	0.1763
55	0.9937	1	0.63%	0.1962
60	1.0405	1	4.05%	0.2255
65	1.0217	1	2.17%	0.2417
70	0.9696	0.9999	3.0360%	0.2586
75	0.9999	0.9996	0.0300%	0.3035
80	0.9869	0.9972	1.0358%	0.3436
85	0.9580	0.9851	2.7588%	0.3942
90	0.9363	0.9456	0.9761%	0.4555
95	0.8421	0.8560	1.6283%	0.4767
100	0.7069	0.7088	0.2736%	0.4754
105	0.5105	0.5247	2.7021%	0.4241
110	0.3247	0.3434	5.4303%	0.3498
115	0.1972	0.1983	0.5395%	0.2669
120	0.1064	0.1014	4.9177%	0.1839
125	0.0471	0.0463	1.7868%	0.1317
130	0.0194	0.0190	2.0990%	0.0775
135	0.0057	0.0071	19.546%	0.0444
140	0.0034	0.0024	39.979%	0.0350
145	0.0007	0.0007	2.93%	0.0203
150	0.0004	0.0002	96.259%	0.0075

Table A.1: Malliavin estimates of the Δ of a vanilla European call option after 20000 simulations. These data are represented in figure 3.2a on page 39.

K	**Mall. Γ**	**B.–S. Γ**	**Error**	**SNR**
50	-0.005	2.e-14	2.e+13%	0.0074
55	-0.003	2.e-11	1.e+10%	0.0042
60	-0.001	4.4e-9	3.4e+7%	0.0022
65	0.0064	2.9e-7	2.1e+6%	0.0097
70	0.0044	8.3e-6	53707.%	0.0078
75	-1.e-4	1.1e-4	228.21%	2.7e-4
80	6.2e-4	8.3e-4	25.065%	0.0013
85	3.1e-4	0.0037	91.585%	7.9e-4
90	0.0076	0.0110	30.342%	0.0217
95	0.0210	0.0226	7.2190%	0.0634
100	0.0336	0.0342	1.8691%	0.1174
105	0.0376	0.0398	5.3243%	0.1645
110	0.0330	0.0367	10.084%	0.1809
115	0.0274	0.0278	1.4575%	0.1731
120	0.0192	0.0177	8.4564%	0.1313
125	0.0096	0.0097	0.1669%	0.1059
130	0.0048	0.0046	3.7657%	0.0614
135	0.0015	0.0019	20.104%	0.0390
140	0.0010	7.5e-4	38.696%	0.0331
145	2.5e-4	2.6e-4	2.4515%	0.0197
150	1.9e-4	8.5e-5	129.04%	0.0074

Table A.2: Malliavin estimates of the Γ of a vanilla European call option after 20000 simulations. These data are represented in figure 3.2b on page 39.

K	**Mall. Δ**	**B.–S. Δ**	**Error**	**SNR**
50	1.2e-4	-4e-14	2.e+11%	0.0090
55	-5.e-4	4.e-11	1.2e+9%	0.0052
60	3.0e-4	7.4e-9	4.1e+6%	0.0032
65	1.2e-4	-2.e-7	56847.%	0.0094
70	4.4e-5	-4.e-6	999.54%	0.0032
75	1.1e-4	-5.e-5	326.72%	0.0085
80	-3.e-4	-2.e-4	5.7652%	0.0243
85	-9.e-4	-9.e-4	5.7206%	0.0805
90	-0.002	-0.001	2.7989%	0.2055
95	-0.002	-0.002	2.0770%	0.2536
100	-0.001	-0.001	2.6137%	0.2019
105	-2.e-4	-2.e-4	14.459%	0.0222
110	0.0013	0.0013	2.8266%	0.1669
115	0.0021	0.0020	3.1269%	0.2692
120	0.0018	0.0018	1.4505%	0.2383
125	0.0013	0.0013	0.4902%	0.1819
130	7.5e-4	7.4e-4	2.0249%	0.1142
135	3.7e-4	3.5e-4	5.3696%	0.0728
140	2.0e-4	1.5e-4	32.475%	0.0478
145	1.1e-4	5.8e-5	91.905%	0.0320
150	0	2.0e-5	100%	—

Table A.3: Malliavin estimates of the Δ of a Binary European call option after 20000 simulations. These data are represented in figure 3.2c on page 39.

K	**Mall. Γ**	**B.–S. Γ**	**Error**	**SNR**
50	-8.e-5	5.e-14	1.e+11%	9.1e-4
55	9.1e-5	-2e-11	3.6e+8%	0.0067
60	5.2e-7	-4.e-9	12556.%	3.9e-5
65	3.7e-4	4.6e-7	81307.%	0.0039
70	-6.e-5	1.1e-5	628.90%	6.6e-4
75	-2.e-4	1.4e-4	291.40%	0.0029
80	0.0018	0.0010	75.096%	0.0195
85	0.0052	0.0044	19.919%	0.0589
90	0.0125	0.0122	2.7174%	0.1563
95	0.0237	0.0238	0.3016%	0.3428
100	0.0342	0.0342	0.1982%	0.5848
105	0.0378	0.0379	0.2056%	0.6806
110	0.0337	0.0334	1.0497%	0.5879
115	0.0243	0.0242	0.5284%	0.4303
120	0.0144	0.0147	2.2123%	0.2924
125	0.0079	0.0077	2.5387%	0.1983
130	0.0035	0.0035	0.5622%	0.1203
135	0.0015	0.0014	5.2398%	0.0749
140	7.1e-4	5.4e-4	32.385%	0.0484
145	3.5e-4	1.8e-4	92.143%	0.0323
150	0	5.7e-5	-100%	—

Table A.4: Malliavin estimates of the Γ of a Binary European call option after 20000 simulations. These data are represented in figure 3.2d on page 39.

		Finite Difference			**Pathwise**			**Malliavin**		
K	**B.–S. Δ**	**Est. Δ**	**Error**	**SNR**	**Est. Δ**	**Error**	**SNR**	**Est. Δ**	**Error**	**SNR**
50	1.00000	0.9592	4.0724%	0.1352	0.9998	0.0142%	9.9929	0.9890	1.0932%	0.1788
55	1.00000	0.9990	0.0982%	0.1406	1.0004	0.0372%	10.001	1.0135	1.3548%	0.1995
60	1.00000	1.0220	2.2025%	0.1442	1.0006	0.0602%	10.016	1.0202	2.0186%	0.2215
65	1.00000	1.0186	1.8627%	0.1438	1.0008	0.0764%	9.8963	1.0419	4.1867%	0.2454
70	0.99998	0.9841	1.5863%	0.1385	0.9998	0.0112%	9.9552	0.9994	0.0549%	0.2689
75	0.99969	1.0688	6.9140%	0.1498	0.9998	0.0132%	9.8029	1.0018	0.2080%	0.3046
80	0.99729	0.9612	3.6193%	0.1360	0.9971	0.0107%	8.5989	0.9908	0.6485%	0.3477
85	0.98519	0.9232	6.2927%	0.1323	0.9857	0.0541%	5.9769	0.9790	0.6199%	0.3974
90	0.94560	0.9221	2.4816%	0.1358	0.9432	0.2537%	3.4920	0.9443	0.1328%	0.4448
95	0.85609	0.8102	5.3522%	0.1293	0.8568	0.0848%	2.1927	0.8506	0.6371%	0.4769
100	0.70884	0.7009	1.1142%	0.1284	0.7100	0.1641%	1.4273	0.7237	2.1019%	0.4751
105	0.52476	0.5010	4.5234%	0.1138	0.5283	0.6773%	0.9733	0.5244	0.0520%	0.4290
110	0.34344	0.3678	7.1097%	0.1116	0.3454	0.5770%	0.6672	0.3500	1.9385%	0.3566
115	0.19832	0.1960	1.1674%	0.0858	0.2005	1.1127%	0.4576	0.2014	1.5792%	0.2658
120	0.10147	0.0962	5.1618%	0.0638	0.1032	1.7436%	0.3067	0.1081	6.5395%	0.1901
125	0.04633	0.0470	1.5695%	0.0516	0.0479	3.4392%	0.2005	0.0481	3.8439%	0.1292
130	0.01905	0.0172	9.5163%	0.0327	0.0188	1.1145%	0.1220	0.0187	1.6953%	0.0829
135	0.00712	0.0095	34.262%	0.0297	0.0074	5.0290%	0.0751	0.0082	16.132%	0.0494
140	0.00244	0.0028	15.005%	0.0182	0.0021	12.746%	0.0393	0.0023	4.3498%	0.0245
145	7.73e-4	0.0012	67.826%	0.0112	0.0006	12.681%	0.0217	7.7e-4	0.2506%	0.0157
150	2.86e-4	0.0005	145.31%	0.0104	0.0002	13.336%	0.0132	3.8e-4	69.892%	0.0108

Table A.5: Comparative estimates of the Δ of a vanilla European call with finite difference, pathwise derivative, and Malliavin methods. Some of these data are presented in figures 3.6a and 3.7a on page 44.

		Finite Difference			**Pathwise**	**Malliavin**		
K	**B.–S. Γ**	**Est. Γ**	**Error**	**SNR**		**Est. Γ**	**Error**	**SNR**
50	2.e-14	6.8e-4	2.e+12%	2.e-05	N/A	-0.001	4.e+12%	0.0013
55	2.e-11	0.0749	3.e+11%	0.0030	N/A	-0.001	5.8e+9%	0.0016
60	4.4e-9	-0.150	3.3e+9%	0.0061	N/A	-0.002	6.3e+7%	0.0041
65	2.9e-7	-0.141	4.7e+7%	0.0057	N/A	0.0088	2.9e+6%	0.0135
70	8.3e-6	-0.088	1.0e+6%	0.0036	N/A	-6.e-5	851.06%	0.0001
75	0.1e-4	-0.107	95306.%	0.0043	N/A	-8.e-5	171.32%	0.0001
80	0.0008	0.0227	2632.5%	0.0009	N/A	-0.001	292.20%	0.0035
85	0.0037	0.0778	1977.8%	0.0032	N/A	0.0028	24.892%	0.0068
90	0.0110	0.2194	1889.8%	0.0092	N/A	0.0130	18.409%	0.0340
95	0.0226	0.0592	161.41%	0.0027	N/A	0.0228	0.8160%	0.0681
100	0.0342	-0.098	386.48%	0.0051	N/A	0.0371	8.2265%	0.1256
105	0.0398	-0.025	164.89%	0.0016	N/A	0.0393	1.2279%	0.1679
110	0.0367	-0.063	273.93%	0.0055	N/A	0.0372	1.1725%	0.1922
115	0.0278	-0.011	140.46%	0.0014	N/A	0.0284	2.1305%	0.1721
120	0.0177	-0.031	279.91%	0.0060	N/A	0.0192	8.5704%	0.1398
125	0.0097	-0.012	229.72%	0.0039	N/A	0.0100	3.7022%	0.1039
130	0.0046	-0.002	164.36%	0.0016	N/A	0.0045	2.9861%	0.0730
135	0.0019	2.9e-4	84.848%	0.0002	N/A	0.0023	19.480%	0.0443
140	0.0007	0.0018	139.54%	0.0031	N/A	0.0007	0.6908%	0.0213
145	2.6e-4	0.0012	379.64%	0.0037	N/A	0.0002	0.6948%	0.0149
150	8.5e-5	-2.e-4	428.78%	0.0013	N/A	0.0001	71.957%	0.0106

Table A.6: Comparative estimates of the Γ of a vanilla European call with finite difference, pathwise derivative, and Malliavin methods.

		Finite Difference			**Pathwise**			**Malliavin**		
K	**B.–S.** $\mathcal{V}$	**Est.** $\mathcal{V}$	**Error**	**SNR**	**Est.** $\mathcal{V}$	**Error**	**SNR**	**Est.** $\mathcal{V}$	**Error**	**SNR**
50	2.e-11	-21.33	7.e+13%	0.0060	-0.164	5.e+12%	0.0016	-1.123	4.e+12%	0.0013
55	2.e-08	-13.99	6.e+10%	0.0039	0.3431	1.5e+9%	0.0034	-1.296	5.8e+9%	0.0016
60	4.e-06	2.9330	6.5e+7%	0.0008	0.5626	1.2e+7%	0.0055	-2.832	6.3e+7%	0.0041
65	0.0009	5.9331	1.9e+6%	0.0016	0.8395	280095%	0.0082	8.8854	2.9e+6%	0.0135
70	0.0083	-10.37	124658%	0.0029	-0.109	1409.7%	0.0010	-0.062	851.06%	0.0001
75	0.1124	5.1095	4444.0%	0.0014	0.2013	79.045%	0.0019	-0.080	171.32%	0.0001
80	0.8336	-16.09	2030.8%	0.0045	0.7555	9.3716%	0.0075	-1.602	292.20%	0.0035
85	3.7453	-15.09	503.03%	0.0043	3.5959	3.9897%	0.0370	2.8130	24.892%	0.0068
90	11.028	33.545	204.18%	0.0098	10.606	3.8281%	0.1164	13.058	18.409%	0.0340
95	22.676	21.652	4.5128%	0.0068	22.393	1.2463%	0.2786	22.861	0.8160%	0.0681
100	34.294	29.531	13.887%	0.0107	34.983	2.0089%	0.4926	37.115	8.2265%	0.1256
105	39.817	42.501	6.7409%	0.0191	39.934	0.2948%	0.6088	39.328	1.2279%	0.1679
110	36.781	26.488	27.984%	0.0160	37.318	1.4600%	0.5545	37.212	1.1725%	0.1922
115	27.854	33.111	18.872%	0.0289	28.192	1.2109%	0.4208	28.448	2.1305%	0.1721
120	17.737	20.896	17.807%	0.0289	18.227	2.7635%	0.2938	19.257	8.5704%	0.1398
125	9.7048	8.4805	12.615%	0.0188	10.047	3.5339%	0.1959	10.064	3.7022%	0.1039
130	4.6471	6.0764	30.757%	0.0230	4.5950	1.1195%	0.1204	4.5083	2.9861%	0.0730
135	1.9787	2.2112	11.754%	0.0156	2.0990	6.0836%	0.0742	2.3641	19.480%	0.0443
140	0.7595	1.0961	44.322%	0.0139	0.6672	12.141%	0.0390	0.7542	0.6908%	0.0213
145	0.2659	0.3863	45.278%	0.0096	0.2348	11.695%	0.0216	0.2678	0.6948%	0.0149
150	0.0858	-0.048	156.14%	0.0027	0.1004	16.989%	0.0131	0.1476	71.957%	0.0106

Table A.7: Comparative estimates of the $\mathcal{V}$ of a vanilla European call with finite difference, pathwise derivative, and Malliavin methods. Some of these data are presented in figures 3.6b and 3.7b on page 44.

	Finite Difference		**Pathwise**		**Malliavin**	
K	**Est.** Δ	**SNR**	**Est.** Δ	**SNR**	**Est.** Δ	**SNR**
50	0.9792	0.2450	0.9994	9.9578	0.9166	0.0902
55	0.9575	0.2414	1.0014	9.9296	1.1025	0.1188
60	0.9666	0.2413	0.9997	9.9860	0.9526	0.1153
65	0.9927	0.2481	0.9996	9.9484	0.9496	0.1292
70	0.9949	0.2482	1.0000	10.009	0.9713	0.1522
75	0.9726	0.2436	0.9998	10.048	0.9547	0.1755
80	0.9940	0.2479	1.0000	9.9698	0.9740	0.2134
85	1.0207	0.2544	0.9988	9.7129	0.9486	0.2587
90	0.9480	0.2388	0.9896	6.6734	0.9575	0.3378
95	0.9079	0.2425	0.9198	2.9659	0.9014	0.4255
100	0.6746	0.2259	0.6882	1.3727	0.6507	0.4457
105	0.3509	0.1847	0.3641	0.7019	0.3455	0.3503
110	0.1200	0.1238	0.1259	0.3486	0.1175	0.2065
115	0.0263	0.0683	0.0299	0.1582	0.0247	0.1024
120	0.0043	0.0329	0.0044	0.0585	0.0036	0.0378
125	0.0005	0.0126	0.0004	0.0186	0.0002	0.0140
130	4.e-06	0.0050	7.e-05	0.0070	5.e-05	0.0070
135	0	—	0	—	0	—
140	0	—	0	—	0	—
145	0	—	0	—	0	—
150	0	—	0	—	0	—

Table A.8: Comparative estimates of the Δ of a vanilla Asian call with finite difference, pathwise derivative, and Malliavin methods.

	Finite Difference		Pathwise	Malliavin	
K	**Est. Γ**	**SNR**		**Est. Γ**	**SNR**
50	-0.029	0.0021	N/A	0.0047	0.0016
55	-0.179	0.0129	N/A	0.0216	0.0081
60	0.0094	0.0006	N/A	-0.004	0.0017
65	0.0344	0.0024	N/A	0.0072	0.0033
70	-0.006	0.0004	N/A	-0.008	0.0047
75	-0.008	0.0006	N/A	-0.015	0.0100
80	-0.066	0.0048	N/A	-0.001	0.0014
85	0.1169	0.0084	N/A	-0.005	0.0045
90	-0.003	0.0002	N/A	0.0023	0.0026
95	-0.067	0.0052	N/A	0.0308	0.0416
100	0.0694	0.0067	N/A	0.0570	0.1061
105	0.0398	0.0061	N/A	0.0633	0.1654
110	0.0442	0.0136	N/A	0.0336	0.1328
115	0.0119	0.0094	N/A	0.0090	0.0824
120	0.0036	0.0087	N/A	0.0017	0.0321
125	0.0009	0.0090	N/A	0.0001	0.0130
130	-0.000	0.0066	N/A	3.e-05	0.0070
135	0	—	N/A	0	—
140	0	—	N/A	0	—
145	0	—	N/A	0	—
150	0	—	N/A	0	—

Table A.9: Comparative estimates of the Γ of a vanilla Asian call with finite difference, pathwise derivative, and Malliavin methods.

		Finite Difference			Pathwise	Malliavin		
K	**B.–S. Δ**	**Est. Δ**	**Error**	**SNR**		**Est. Δ**	**Error**	**SNR**
50	5.e-14	0	100%	—	N/A	-4.e-5	7.e+10%	4.2e-4
55	4.e-11	0	100%	—	N/A	-0.001	3.0e+9%	0.0127
60	7.4e-9	0	100%	—	N/A	-0.001	1.6e+7%	0.0131
65	4.6e-7	0	100%	—	N/A	1.8e-4	40624.%	0.0019
70	1.1e-5	0	100%	0	N/A	9.3e-4	7737.6%	0.0098
75	1.4e-4	1.0e-4	28.620%	0.0070	N/A	3.1e-4	109.94%	0.0033
80	0.0010	9.9e-4	4.1507%	0.0241	N/A	0.0024	132.37%	0.0259
85	0.0044	0.0047	6.8613%	0.0513	N/A	0.0048	11.190%	0.0545
90	0.0122	0.0133	8.7773%	0.0791	N/A	0.0118	3.5916%	0.1461
95	0.0238	0.0248	3.9111%	0.0987	N/A	0.0235	1.2719%	0.3437
100	0.0342	0.0338	1.2557%	0.1076	N/A	0.0338	1.3291%	0.5793
105	0.0379	0.0380	0.3371%	0.1132	N/A	0.0375	0.8492%	0.6801
110	0.0334	0.0348	4.0843%	0.1123	N/A	0.0335	0.3709%	0.5900
115	0.0242	0.0242	0.0033%	0.0949	N/A	0.0242	0.0653%	0.4332
120	0.0147	0.0154	4.4128%	0.0829	N/A	0.0148	0.5603%	0.2970
125	0.0077	0.0083	7.9714%	0.0652	N/A	0.0076	0.8414%	0.1935
130	0.0035	0.0036	3.1151%	0.0452	N/A	0.0035	0.3725%	0.1210
135	0.0014	0.0014	1.8379%	0.0290	N/A	0.0014	3.5345%	0.0719
140	5.4e-4	4.9e-4	7.9451%	0.0180	N/A	9.5e-4	12.366%	0.0445
145	1.8e-4	1.1e-4	35.175%	0.0077	N/A	1.6e-4	10.759%	0.0223
150	5.7e-5	5.9e-5	3.8663%	0.0094	N/A	1.8e-5	67.409%	0.0070

Table A.10: Comparative estimates of the Δ of a binary European call with finite difference, pathwise derivative, and Malliavin methods.

		Finite Difference			Pathwise	Malliavin		
K	**B.–S.** Γ	**Est.** Γ	**Error**	**SNR**		**Est.** Γ	**Error**	**SNR**
50	-4e-14	0	100%	—	N/A	6.2e-5	1.e+11%	0.0045
55	-2.-11	0	100%	—	N/A	4.2e-5	1.6e+8%	0.0031
60	-4.e-9	0	100%	—	N/A	1.5e-4	3.6e+6%	0.0111
65	-2.e-7	0	100%	—	N/A	9.9e-5	44348.%	0.0072
70	-4.e-6	0	100%	0	N/A	0.1e-4	1965.0%	0.0076
75	-5.e-5	4.3e-4	979.46%	0.0074	N/A	-1.e-4	226.09%	0.0127
80	-2.e-4	-6.e-4	113.32%	0.0044	N/A	-3.e-4	6.7301%	0.0246
85	-9.e-4	0.0025	363.00%	0.0077	N/A	-8.e-4	10.271%	0.0745
90	-0.001	0.0027	241.59%	0.0047	N/A	-0.001	0.3568%	0.1960
95	-0.002	-0.001	26.891%	0.0021	N/A	-0.002	3.0476%	0.2775
100	-0.001	0.0066	453.01%	0.0060	N/A	-0.001	0.7061%	0.1991
105	-2.e-4	0.0064	2826.6%	0.0055	N/A	-2.e-4	17.631%	0.0302
110	0.0013	-0.003	362.88%	0.0032	N/A	0.0013	3.2835%	0.1653
115	0.0020	0.0045	121.23%	0.0051	N/A	0.0020	1.9025%	0.2657
120	0.0018	-0.002	241.52%	0.0040	N/A	0.0019	1.3039%	0.2434
125	0.0013	0.0022	69.415%	0.0049	N/A	0.0012	0.7867%	0.1759
130	7.4e-4	3.3e-4	55.085%	0.0011	N/A	7.4e-4	0.7848%	0.1147
135	3.5e-4	4.5e-4	25.773%	0.0026	N/A	3.4e-4	4.1239%	0.0701
140	1.5e-4	-5.e-4	473.76%	0.0056	N/A	1.7e-4	12.988%	0.0430
145	5.8e-5	4.7e-5	18.089%	9.0e-4	N/A	5.0e-5	13.792%	0.0222
150	2.0e-5	7.1e-5	255.64%	0.0038	N/A	6.6e-6	66.745%	0.0070

Table A.11: Comparative estimates of the Γ of a binary European call with finite difference, pathwise derivative, and Malliavin methods.

		Finite Difference			Pathwise	Malliavin		
K	**B.–S.** $\mathcal{V}$	**Est.** $\mathcal{V}$	**Error**	**SNR**		**Est.** $\mathcal{V}$	**Error**	**SNR**
50	-4e-11	0	100%	—	N/A	0.0622	1.e+11%	0.0045
55	-2.e-8	0	100%	—	N/A	0.0425	1.6e+8%	0.0031
60	-4.e-6	0	100%	—	N/A	0.1523	3.6e+6%	0.0111
65	-2.e-4	0	100%	—	N/A	0.0990	44348.%	0.0072
70	-0.004	0	100%	—	N/A	-0.101	1965.0%	0.0076
75	-0.051	-0.017	65.284%	0.0022	N/A	-0.167	226.09%	0.0127
80	-0.289	-0.261	9.7467%	0.0125	N/A	-0.309	6.7301%	0.0246
85	-0.958	-0.939	1.9934%	0.0208	N/A	-0.860	10.271%	0.0745
90	-1.965	-1.914	2.5771%	0.0227	N/A	-1.958	0.3568%	0.1960
95	-2.537	-2.901	14.348%	0.0230	N/A	-2.614	3.0476%	0.2775
100	-1.886	-0.909	51.775%	0.0057	N/A	-1.899	0.7061%	0.1991
105	-0.235	-0.214	9.1125%	0.0012	N/A	-0.277	17.631%	0.0302
110	1.3479	1.8846	39.822%	0.0122	N/A	1.3036	3.2835%	0.1653
115	2.0531	2.1819	6.2746%	0.0173	N/A	2.0140	1.9025%	0.2657
120	1.8820	1.6409	12.812%	0.0175	N/A	1.9065	1.3039%	0.2434
125	1.3054	1.4031	7.4784%	0.0219	N/A	1.2952	0.7867%	0.1759
130	0.7412	0.434	41.451%	0.0106	N/A	0.7470	0.7848%	0.1147
135	0.3592	0.4696	30.738%	0.0188	N/A	0.3444	4.1239%	0.0701
140	0.1527	0.1664	9.0159%	0.0125	N/A	0.1725	12.988%	0.0430
145	0.0580	0.0416	28.328%	0.0045	N/A	0.0500	13.792%	0.0222
150	0.0200	0.0237	18.546%	0.0053	N/A	0.0066	66.745%	0.0070

Table A.12: Comparative estimates of the $\mathcal{V}$ of a binary European call with finite difference, pathwise derivative, and Malliavin methods.

	Finite Difference		**Pathwise**	**Malliavin**	
K	**Est.** Δ	**SNR**		**Est.** Δ	**SNR**
50	0	—	N/A	-1.e-4	5.3e-4
55	0	—	N/A	-0.002	0.0127
60	0	—	N/A	-0.002	0.0130
65	0	—	N/A	3.0e-4	0.0015
70	0	—	N/A	0.0018	0.0095
75	0	—	N/A	5.8e-5	3.0e-4
80	1.1e-5	0.0050	N/A	0.0028	0.0152
85	3.3e-4	0.0194	N/A	8.0e-4	0.0042
90	0.0059	0.0756	N/A	0.0037	0.0205
95	0.0284	0.1402	N/A	0.0274	0.1718
100	0.0607	0.1911	N/A	0.0594	0.4713
105	0.0577	0.1837	N/A	0.0586	0.5232
110	0.0298	0.1428	N/A	0.0297	0.3177
115	0.0090	0.0862	N/A	0.0086	0.1469
120	0.0015	0.0385	N/A	0.0017	0.0589
125	2.2e-4	0.0165	N/A	1.9e-4	0.0175
130	-1.e-5	0.0028	N/A	3.2e-5	0.0070
135	0	—	N/A	0	—
140	0	—	N/A	0	—
145	0	—	N/A	0	—
150	0	—	N/A	0	—

Table A.13: Comparative estimates of the Δ of a binary Asian call with finite difference, pathwise derivative, and Malliavin methods.

	Finite Difference		**Pathwise**	**Malliavin**	
K	**Est.** Γ	**SNR**		**Est.** Γ	**SNR**
50	0	—	N/A	2.5e-4	0.0045
55	0	—	N/A	1.4e-4	0.0027
60	0	—	N/A	6.0e-4	0.0110
65	0	—	N/A	3.7e-4	0.0068
70	0	—	N/A	-3.e-4	0.0065
75	0	—	N/A	-3.e-4	0.0064
80	-2.e-5	0.0050	N/A	-9.e-5	0.0016
85	4.7e-5	7.9e-4	N/A	0.0005	0.0091
90	-0.001	0.0067	N/A	-0.001	0.0405
95	-0.010	0.0148	N/A	-0.006	0.1671
100	-0.001	0.0017	N/A	-0.004	0.1243
105	0.0147	0.0134	N/A	0.0039	0.1125
110	0.0015	0.0021	N/A	0.0058	0.1996
115	0.0033	0.0094	N/A	0.0027	0.1210
120	2.8e-4	0.0020	N/A	7.8e-4	0.0536
125	1.6e-4	0.0037	N/A	1.1e-4	0.0162
130	-2.e-5	0.0015	N/A	2.3e-5	0.0069
135	0	—	N/A	0	—
140	0	—	N/A	0	—
145	0	—	N/A	0	—
150	0	—	N/A	0	—

Table A.14: Comparative estimates of the Γ of a binary Asian call with finite difference, pathwise derivative, and Malliavin methods.

APPENDIX B

Source Code

Throughout this work, a number of `MATLAB` scripts are explicitly referenced. For the sake of completeness, this appendix contains the source code of those `.m` files.

The following `MATLAB` code performed the Monte Carlo simulation computations in example 1.1.1. The curve–fitting in that example used the `MATLAB` curve–fitting toolbox with a eigth–order polynomial.

File Fitting.m

```
function Fitting (NumSimulations)

S0base = 100;
r = 0.05;
sigma = 0.1;
T = 1;
K = 100

epsilon = 0.1;

S0 = [S0base - 4*epsilon:epsilon:S0base+4*epsilon];

[S] = GBMPaths(S0(5),r,sigma,T,NumSimulations);
[S1] = GBMPaths(S0(6),r,sigma,T,NumSimulations);
[S2] = GBMPaths(S0(7),r,sigma,T,NumSimulations);
[S3] = GBMPaths(S0(8),r,sigma,T,NumSimulations);
[S4] = GBMPaths(S0(9),r,sigma,T,NumSimulations);
[SA] = GBMPaths(S0(1),r,sigma,T,NumSimulations);
[SB] = GBMPaths(S0(2),r,sigma,T,NumSimulations);
[SC] = GBMPaths(S0(3),r,sigma,T,NumSimulations);
[SD] = GBMPaths(S0(4),r,sigma,T,NumSimulations);

ST = S(size(S,1),:);
ST1 = S1(size(S1,1),:);
ST2 = S2(size(S2,1),:);
ST3 = S3(size(S3,1),:);
ST4 = S4(size(S4,1),:);
```

```
STA = SA(size(SA,1),:);
STB = SB(size(SB,1),:);
STC = SC(size(SC,1),:);
STD = SD(size(SD,1),:);

Value(5) = exp(-r*T)*sum(max(ST - K,0))/NumSimulations;
Value(6) = exp(-r*T)*sum(max(ST1 - K,0))/NumSimulations;
Value(7) = exp(-r*T)*sum(max(ST2 - K,0))/NumSimulations;
Value(8) = exp(-r*T)*sum(max(ST3 - K,0))/NumSimulations;
Value(9) = exp(-r*T)*sum(max(ST4 - K,0))/NumSimulations;
Value(1) = exp(-r*T)*sum(max(STA - K,0))/NumSimulations;
Value(2) = exp(-r*T)*sum(max(STB - K,0))/NumSimulations;
Value(3) = exp(-r*T)*sum(max(STC - K,0))/NumSimulations;
Value(4) = exp(-r*T)*sum(max(STD - K,0))/NumSimulations;

save 'CurveFitting' S0 Value;
```

The next two scripts are wrappers. They call `GBMPaths.m` to create sample paths and then call `GBM[OptionStyle][ComputationMethod]` to produce estimates of Δ, Γ, and $\mathcal{V}$.

File MasterGBMAsian.m

```
function MasterGBMAsian (PayoffType,K,S0,r,sigma,T,NumSimulations);

disp([ 'PayoffType = ' num2str(PayoffType) ', K = ' num2str(K) ', S0 = ' num2str(S0) ', r = ' ...
   num2str(r) ', sigma = ' num2str(sigma) ',  T = ' num2str(T) ', N = ' num2str(NumSimulations) 10 ]);

DeltaEpsilon = 1;
VegaEpsilon = .002;

Save = 1;

[S,W] = GBMPaths (S0, r, sigma, T, NumSimulations);
SUp = GBMPaths (S0+DeltaEpsilon, r, sigma, T, NumSimulations);
SDown = GBMPaths (S0-DeltaEpsilon, r, sigma, T, NumSimulations);
SSigmaUp = GBMPaths (S0, r, sigma+VegaEpsilon, T, NumSimulations);
SSigmaDown = GBMPaths (S0, r, sigma-VegaEpsilon, T, NumSimulations);

[FiniteDelta RunningFiniteDelta FiniteDeltaSNR FiniteGamma RunningFiniteGamma FiniteGammaSNR ...
   FiniteVega RunningFiniteVega FiniteVegaSNR] = GBMAsianFinite( PayoffType ,S,SUp,SDown, SSigmaUp, ...
   SSigmaDown, K, r, sigma, T, DeltaEpsilon, VegaEpsilon);

[PathwiseDelta RunningPathwiseDelta PathwiseDeltaSNR PathwiseGamma RunningPathwiseGamma ...
   PathwiseGammaSNR PathwiseVega RunningPathwiseVega PathwiseVegaSNR] = GBMAsianPathwise( ...
   PayoffType ,S, W, K, r,sigma, T );

[MalliavinDelta RunningMalliavinDelta MalliavinDeltaSNR MalliavinGamma RunningMalliavinGamma ...
   MalliavinGammaSNR MalliavinVega RunningMalliavinVega MalliavinVegaSNR] = GBMAsianMalliavin(
   PayoffType ,S, W, K, r,sigma, T );

if (PayoffType == 'V')
ActualDelta = 0;% blsdelta(S0,K,r,T,sigma);
ActualGamma = 0;% blsgamma(S0,K,r,T,sigma);
ActualVega = 0;% blsvega(S0,K,r,T,sigma);
end
```

```
if (PayoffType == 'B')
% Implement analytic Black-Scholes greeks for binary option
d1 = 0;% (log(S0/K) + (r + ((sigma^2)/2)*T))/(sigma*sqrt(T));
d2 = 0;% d1 - sigma*sqrt(T);
NPrimed2 = 0;% 1/(sqrt(2*pi)) * exp(-0.5*d2^2);
ActualDelta = 0;% exp(-r*T) * NPrimed2 / (sigma * S0 * sqrt(T));
ActualGamma = 0;% -exp(-r*T) * d1 * NPrimed2 / (sigma^2 * S0^2 * T);
ActualVega = 0;% -exp(-r*T) * NPrimed2 * (d1/sigma);
end

% Print out results
Method = ['Asian Finite ' PayoffType];
Printer( Method, FiniteDelta, ActualDelta, FiniteDeltaSNR, FiniteGamma, ActualGamma, FiniteGammaSNR, ...
   FiniteVega, ActualVega, FiniteVegaSNR )

Method = ['Asian Pathwise ' PayoffType];
Printer( Method, PathwiseDelta, ActualDelta, PathwiseDeltaSNR, PathwiseGamma, ActualGamma, ...
   PathwiseGammaSNR, PathwiseVega, ActualVega, PathwiseVegaSNR )

Method = ['Asian Malliavin ' PayoffType];
Printer( Method, MalliavinDelta, ActualDelta, MalliavinDeltaSNR, MalliavinGamma, ActualGamma, ...
   MalliavinGammaSNR, MalliavinVega, ActualVega, MalliavinVegaSNR )



if Save == 1
Filename = ['Data/MasterGBMAsian,' num2str(PayoffType) ',' num2str(K) ',' num2str(S0) ',' ...
    num2str(r) ',' num2str(sigma) ',' num2str(T) ',' num2str(NumSimulations) '.mat' ];
save(Filename,'PayoffType','K','S0','r','sigma','T','NumSimulations','DeltaEpsilon', ...
   'VegaEpsilon','FiniteDelta','ActualDelta','FiniteDeltaSNR','FiniteGamma','ActualGamma', ...
   'FiniteGammaSNR','FiniteVega','ActualVega','FiniteVegaSNR','PathwiseDelta','ActualDelta', ...
   'PathwiseDeltaSNR','PathwiseGamma','ActualGamma','PathwiseGammaSNR','PathwiseVega', ...
   'ActualVega','PathwiseVegaSNR','MalliavinDelta','ActualDelta','MalliavinDeltaSNR', ...
   'MalliavinGamma','ActualGamma','MalliavinGammaSNR','MalliavinVega','ActualVega', ...
   'MalliavinVegaSNR','RunningFiniteDelta','RunningFiniteGamma','RunningFiniteVega', ...
  'RunningPathwiseDelta','RunningPathwiseGamma','RunningPathwiseVega','RunningMalliavinDelta', ...
  'RunningMalliavinGamma','RunningMalliavinVega');
end

end
```

File MasterGBMEuro.m

```
function MasterGBMEuro (PayoffType,K,S0,r,sigma,T,NumSimulations);

disp([ 'PayoffType = ' num2str(PayoffType) ', K = ' num2str(K) ', S0 = ' num2str(S0) ', r = ' num2str(r) ...
   ', sigma = ' num2str(sigma) ', T = ' num2str(T) ', N = ' num2str(NumSimulations) 10 ]);

DeltaEpsilon = 1;
VegaEpsilon = .002;

% Boolean: should we plot [Delta Gamma Vega]?
MakePlots = [0 0 0];
Save = 1;
```

```
[S,W] = GBMPaths (S0, r, sigma, T, NumSimulations);
SUp = GBMPaths (S0+DeltaEpsilon, r, sigma, T, NumSimulations);
SDown = GBMPaths (S0-DeltaEpsilon, r, sigma, T, NumSimulations);
SSigmaUp = GBMPaths (S0, r, sigma+VegaEpsilon, T, NumSimulations);
SSigmaDown = GBMPaths (S0, r, sigma-VegaEpsilon, T, NumSimulations);

[FiniteDelta RunningFiniteDelta FiniteDeltaSNR FiniteGamma RunningFiniteGamma FiniteGammaSNR ...
   FiniteVega RunningFiniteVega FiniteVegaSNR] = GBMEuroFinite( PayoffType ,S,SUp,SDown, ...
   SSigmaUp, SSigmaDown, K, r, sigma, T, DeltaEpsilon, VegaEpsilon);

[PathwiseDelta RunningPathwiseDelta PathwiseDeltaSNR PathwiseGamma RunningPathwiseGamma ...
   PathwiseGammaSNR PathwiseVega RunningPathwiseVega PathwiseVegaSNR] = GBMEuroPathwise( ...
   PayoffType ,S, W, K, r,sigma, T );

[MalliavinDelta RunningMalliavinDelta MalliavinDeltaSNR MalliavinGamma RunningMalliavinGamma ...
   MalliavinGammaSNR MalliavinVega RunningMalliavinVega MalliavinVegaSNR] = GBMEuroMalliavin( ...
  PayoffType ,S, W, K, r,sigma, T );

if (PayoffType == 'V')
 ActualDelta = blsdelta(S0,K,r,T,sigma);
 ActualGamma = blsgamma(S0,K,r,T,sigma);
 ActualVega = blsvega(S0,K,r,T,sigma);
end

if (PayoffType == 'B')
% Analytic B-S greeks for binary European option
d1 = (log(S0/K) + (r + ((sigma^2)/2)*T))/(sigma*sqrt(T));
d2 = d1 - sigma*sqrt(T);
NPrimed2 = 1/(sqrt(2*pi)) * exp(-0.5*d2^2);
ActualDelta = exp(-r*T) * NPrimed2 / (sigma * S0 * sqrt(T));
ActualGamma = -exp(-r*T) * d1 * NPrimed2 / (sigma^2 * S0^2 * T);
ActualVega = -exp(-r*T) * NPrimed2 * (d1/sigma);
end

% Print out results
Method = ['Euro Finite ' PayoffType];
Printer( Method, FiniteDelta, ActualDelta, FiniteDeltaSNR, FiniteGamma, ActualGamma, ...
   FiniteGammaSNR, FiniteVega, ActualVega, FiniteVegaSNR )

Method = ['Euro Pathwise ' PayoffType];
Printer( Method, PathwiseDelta, ActualDelta, PathwiseDeltaSNR, PathwiseGamma, ActualGamma, ...
   PathwiseGammaSNR, PathwiseVega, ActualVega, PathwiseVegaSNR )

Method = ['Euro Malliavin ' PayoffType];
Printer( Method, MalliavinDelta, ActualDelta, MalliavinDeltaSNR, MalliavinGamma, ActualGamma, ...
   MalliavinGammaSNR, MalliavinVega, ActualVega, MalliavinVegaSNR )

if Save == 1
Filename = ['Data/MasterGBMEuro,' num2str(PayoffType) ',' num2str(K) ',' num2str(S0) ',' ...
   num2str(r) ',' num2str(sigma) ',' num2str(T) ',' num2str(NumSimulations) '.mat' ];
save(Filename,'PayoffType','K','S0','r','sigma','T','NumSimulations','DeltaEpsilon','VegaEpsilon', ...
  'FiniteDelta','ActualDelta','FiniteDeltaSNR','FiniteGamma','ActualGamma','FiniteGammaSNR', ...
   'FiniteVega','ActualVega','FiniteVegaSNR','PathwiseDelta','ActualDelta','PathwiseDeltaSNR', ...
   'PathwiseGamma','ActualGamma','PathwiseGammaSNR','PathwiseVega','ActualVega','PathwiseVegaSNR', ...
   'MalliavinDelta','ActualDelta','MalliavinDeltaSNR','MalliavinGamma','ActualGamma','MalliavinGammaSNR', ...
   'MalliavinVega','ActualVega','MalliavinVegaSNR','RunningFiniteDelta','RunningFiniteGamma', ...
   'RunningFiniteVega','RunningPathwiseDelta','RunningPathwiseGamma','RunningPathwiseVega', ...
   'RunningMalliavinDelta','RunningMalliavinGamma','RunningMalliavinVega');
end

end
```

This script creates a geometric Brownian motion for use in Monte Carlo simulation.

File GBMPaths.m

```
function [S,W] = GBMPaths (S0, r, sigma, T, NumSimulations)

HeadString = ['Generating GBM paths (S0 = ' num2str(S0) ', r = ' num2str(r) ', sigma = ' ...
   num2str(sigma) ', T = ' num2str(T) ') ...'];
disp(HeadString)

tic

% Calibrate time
N = floor(sqrt(NumSimulations)); % Number of intervals for simulation.
dt = T/N; t = 0:dt:T; length(t); % NB t is length N+1.

% Number of Simulations
tBig = t'*ones(1,NumSimulations);

% GBM calibration
mu = r - (sigma^2)/2;

% Create Weiner processes
dW = randn(N,NumSimulations)*sqrt(dt); W = [zeros(1,NumSimulations); cumsum(dW,1)];

% Create stock prices
S = S0*exp(mu.*tBig + sigma.*W);

toc

TailString = [num2str(NumSimulations) ' Paths generated.' 10];
disp(TailString)

end
```

The following six scripts are all name `GBM[OptionStyle][ComputationMethod]` where `OptionStyle` denotes whether we have a European or Asian option and `ComputationMethod` denotes the estimator (e.g. finite difference) that we are using. Each can be invoked for a binary or a vanilla option.

File GBMEuroFinite.m

```
function [ Delta RunningDelta DeltaSNR Gamma RunningGamma GammaSNR Vega RunningVega VegaSNR ] = ...
   GBMEuroFinite( Payoff, S, SUp,SDown, SSigmaUp, SSigmaDown, K, r, sigma, T, DeltaEpsilon, VegaEpsilon)

% This function uses the cent'l finite diff method to determine sensitivities.
% Outputs for each greek
%        1) estimated sensitivities
%         2) running estimates of sensitivities as simulation progresses.
%         3) signal to noise ratios of the estimators.

% Note: size(S,1) = # of discretized time steps in each simulation
% Initialize
S0 = S(1,1);
ST = S(size(S,1),:);

% Pull in value matrices
STUp = SUp(size(SUp,1),:);
STDown = SDown(size(SDown,1),:);
STSigmaUp = SSigmaUp(size(SSigmaUp,1),:);
STSigmaDown = SSigmaDown(size(SSigmaDown,1),:);

% Payoffs for each GBM path.
if (Payoff == 'B')
Payoff = ST > K;
PayoffUp = STUp > K;
PayoffDown = STDown > K;
PayoffSigmaUp = STSigmaUp > K;
PayoffSigmaDown = STSigmaDown > K;
else
Payoff = max(ST - K,0);
PayoffUp = max(STUp - K,0);
PayoffDown = max(STDown - K,0);
PayoffSigmaUp = max(STSigmaUp - K,0);
PayoffSigmaDown = max(STSigmaDown - K,0);
end

% Valuation for each GBM path.
RealizationValue = (exp(-r*T)*Payoff);
RealizationValueUp = (exp(-r*T)*PayoffUp);
RealizationValueDown = (exp(-r*T)*PayoffDown);
RealizationValueSigmaUp = (exp(-r*T)*PayoffSigmaUp);
RealizationValueSigmaDown = (exp(-r*T)*PayoffSigmaDown);

% Finite-diff method sensitivities for each GBM path.
RealizationForwardDelta = (RealizationValueUp - RealizationValue)/DeltaEpsilon;
RealizationBackwardDelta = (RealizationValue - RealizationValueDown)/DeltaEpsilon;
RealizationDelta = (RealizationValueUp - RealizationValueDown)/(2*DeltaEpsilon);
RealizationGamma = (RealizationForwardDelta - RealizationBackwardDelta)/(DeltaEpsilon);
RealizationForwardVega = (RealizationValueSigmaUp - RealizationValue)/VegaEpsilon;
RealizationBackwardVega = (RealizationValue - RealizationValueSigmaDown)/VegaEpsilon;
RealizationVega = (RealizationValueSigmaUp - RealizationValueSigmaDown)/(2*VegaEpsilon);

% Compute running prices
% Note: size(S,2) = # of simulations in our simulation
RunningPrice = cumsum(exp(-r*T)*Payoff)./cumsum(ones(1,size(S,2)));
RunningPriceUp = cumsum(exp(-r*T)*PayoffUp)./cumsum(ones(1,size(S,2)));
RunningPriceDown = cumsum(exp(-r*T)*PayoffDown)./cumsum(ones(1,size(S,2)));
RunningPriceSigmaUp = cumsum(exp(-r*T)*PayoffSigmaUp)./cumsum(ones(1,size(S,2)));
RunningPriceSigmaDown = cumsum(exp(-r*T)*PayoffSigmaDown)./cumsum(ones(1,size(S,2)));

% Compute running greeks
RunningForwardDelta = (RunningPriceUp - RunningPrice)/DeltaEpsilon;
RunningBackwardDelta = (RunningPrice - RunningPriceDown)/DeltaEpsilon;
```

```
RunningDelta = (RunningForwardDelta + RunningBackwardDelta)/2;
RunningGamma = (RunningForwardDelta - RunningBackwardDelta)/(DeltaEpsilon);
RunningForwardVega = (RunningPriceSigmaUp - RunningPrice)/VegaEpsilon;
RunningBackwardVega = (RunningPrice - RunningPriceSigmaDown)/VegaEpsilon;
RunningVega = (RunningForwardVega + RunningBackwardVega)/2;

% Identify greek as end of running series
% % ForwardDelta = RunningForwardDelta(length(RunningForwardDelta));
% % BackwardDelta = RunningBackwardDelta(length(RunningBackwardDelta));
Delta = RunningDelta(length(RunningDelta));
Gamma = RunningGamma(length(RunningGamma));
% % ForwardVega = RunningForwardVega(length(RunningForwardVega));
% % BackwardVega = RunningBackwardVega(length(RunningBackwardVega));
Vega = RunningVega(length(RunningVega));

DeltaSNR = abs(Delta/std(RealizationDelta));
GammaSNR = abs(Gamma/std(RealizationGamma));
VegaSNR = abs(Vega/std(RealizationVega));

end
```

File GBMEuroMalliavin.m

```
function [ Delta RunningDelta DeltaSNR Gamma RunningGamma GammaSNR Vega RunningVega VegaSNR ] = ...
   GBMEuroMalliavin( PayoffStructure, S, W, K, r,sigma, T )

% This function uses the malliavin method to determine sensitivities.
% Outputs for each greek
%         1) estimated sensitivities
%         2) running estimates of sensitivities as simulation progresses.
%         3) signal to noise ratios of the estimators.

% Initialize
S0 = S(1,1);
ST = S(size(S,1),:);
WT = W(size(W,1),:);

% The payoffs under each realization in S
if (PayoffStructure == 'B')
        Payoff = ST > K;
else
        Payoff = max(ST - K,0);
end

% The computed Greeks for each realization
RealizationDelta = (exp(-r*T)/(sigma*S0*T))*Payoff.*WT;
RealizationGamma = (exp(-r*T)/(sigma*(S0^2)*T))*Payoff.*((WT.^2 - T)/(sigma*T) - WT);
RealizationVega = (exp(-r*T))*Payoff.*((WT.^2 - T)/(sigma*T) - WT);

% The running average Greeks
RunningDelta = cumsum(RealizationDelta)./cumsum(ones(1,size(S,2)));
RunningGamma = cumsum(RealizationGamma)./cumsum(ones(1,size(S,2)));
RunningVega = cumsum(RealizationVega)./cumsum(ones(1,size(S,2)));

% The computed average Greeks
Delta = RunningDelta(length(RunningDelta));
```

```
Gamma = RunningGamma(length(RunningGamma));
Vega = RunningVega(length(RunningVega));

DeltaSNR = abs(Delta/std(RealizationDelta));
GammaSNR = abs(Gamma/std(RealizationGamma));
VegaSNR = abs(Vega/std(RealizationVega));

end
```

File GBMEuroPathwise.m

```
function [ Delta RunningDelta DeltaSNR Gamma RunningGamma GammaSNR Vega RunningVega VegaSNR ] = ...
   GBMEuroPathwise ( PayoffStructure , S, W, K, r,sigma, T )

% This function uses the pathwise derivative method to determine sensitivities.
% Outputs for each greek
%         1) estimated sensitivities
%         2) running estimates of sensitivities as simulation progresses.
%         3) signal to noise ratios of the estimators.

% Initialize
S0 = S(1,1);
ST = S(size(S,1),:);
WT = W(size(W,1),:);

% The payoffs under each realization in S
if (PayoffStructure  == 'B')
        PayoffDeriv = 0;
        PayoffSecondDeriv = 0;
else
        PayoffDeriv = ST > K;
        PayoffSecondDeriv = 0;
end

% The computed Greeks for each realization
RealizationDelta = (exp(-r*T)/S0)*PayoffDeriv.*ST;
RealizationGamma = (exp(-r*T)/S0^2)*PayoffSecondDeriv.*(ST.^2);
RealizationVega = exp(-r*T)*PayoffDeriv.*ST.*(WT - sigma*T);

% The running average Greeks
RunningDelta = cumsum(RealizationDelta)./cumsum(ones(1,size(S,2)));
RunningGamma = cumsum(RealizationGamma)./cumsum(ones(1,size(S,2)));
RunningVega = cumsum(RealizationVega)./cumsum(ones(1,size(S,2)));

% The computed average Greeks
Delta = RunningDelta(length(RunningDelta));
Gamma = RunningGamma(length(RunningGamma));
Vega = RunningVega(length(RunningVega));

% Identify greek as end of running series
% % ForwardDelta = RunningForwardDelta(length(RunningForwardDelta));
% % BackwardDelta = RunningBackwardDelta(length(RunningBackwardDelta));
Delta = RunningDelta(length(RunningDelta));
Gamma = RunningGamma(length(RunningGamma));
% % ForwardVega = RunningForwardVega(length(RunningForwardVega));
% % BackwardVega = RunningBackwardVega(length(RunningBackwardVega));
```

```
Vega = RunningVega(length(RunningVega));

DeltaSNR = abs(Delta/std(RealizationDelta));
GammaSNR = abs(Gamma/std(RealizationGamma));
VegaSNR = abs(Vega/std(RealizationVega));

end
```

File GBMAsianFinite.m

```
function [ Delta RunningDelta DeltaSNR Gamma RunningGamma GammaSNR Vega RunningVega VegaSNR ] = ...
   GBMAsianFinite( Payoff, S, SUp,SDown, SSigmaUp, SSigmaDown, K, r, sigma, T, DeltaEpsilon, VegaEpsilon)

% This function uses the cent'l finite diff method to determine sensitivities.
% Outputs for each greek
%         1) estimated sensitivities
%         2) running estimates of sensitivities as simulation progresses.
%         3) signal to noise ratios of the estimators.

RunningDumbGamma  = 0;

% NB: size(S,1) = # of discretized time steps in each simulation
%     size(S,2) = # of simulations

STBar = sum(S,1)/size(S,1);
STUpBar = sum(SUp,1)/size(SUp,1);
STDownBar = sum(SDown,1)/size(SDown,1);
STSigmaUpBar = sum(SSigmaUp,1)/size(SSigmaUp,1);
STSigmaDownBar = sum(SSigmaDown,1)/size(SSigmaDown,1);

% Payoffs for each GBM path.
if (Payoff == 'B')
Payoff = STBar > K;
PayoffUp = STUpBar > K;
PayoffDown = STDownBar > K;
PayoffSigmaUp = STSigmaUpBar > K;
PayoffSigmaDown = STSigmaDownBar > K;
else
Payoff = max(STBar - K,0);
PayoffUp = max(STUpBar - K,0);
PayoffDown = max(STDownBar - K,0);
PayoffSigmaUp = max(STSigmaUpBar - K,0);
PayoffSigmaDown = max(STSigmaDownBar - K,0);
end

% Valuation for each GBM path.
RealizationValue = (exp(-r*T)*Payoff);
RealizationValueUp = (exp(-r*T)*PayoffUp);
RealizationValueDown = (exp(-r*T)*PayoffDown);
RealizationValueSigmaUp = (exp(-r*T)*PayoffSigmaUp);
RealizationValueSigmaDown = (exp(-r*T)*PayoffSigmaDown);

% Finite-diff method sensitivities for each GBM path.
RealizationForwardDelta = (RealizationValueUp - RealizationValue)/DeltaEpsilon;
RealizationBackwardDelta = (RealizationValue - RealizationValueDown)/DeltaEpsilon;
RealizationDelta = (RealizationValueUp - RealizationValueDown)/(2*DeltaEpsilon);
```

```
RealizationGamma = (RealizationForwardDelta - RealizationBackwardDelta)/(DeltaEpsilon);
RealizationForwardVega = (RealizationValueSigmaUp - RealizationValue)/VegaEpsilon;
RealizationBackwardVega = (RealizationValue - RealizationValueSigmaDown)/VegaEpsilon;
RealizationVega = (RealizationValueSigmaUp - RealizationValueSigmaDown)/(2*VegaEpsilon);

% Compute running prices
% Note: size(S,2) = # of simulations in our simulation
RunningPrice = cumsum(exp(-r*T)*Payoff)./cumsum(ones(1,size(S,2)));
RunningPriceUp = cumsum(exp(-r*T)*PayoffUp)./cumsum(ones(1,size(S,2)));
RunningPriceDown = cumsum(exp(-r*T)*PayoffDown)./cumsum(ones(1,size(S,2)));
RunningPriceSigmaUp = cumsum(exp(-r*T)*PayoffSigmaUp)./cumsum(ones(1,size(S,2)));
RunningPriceSigmaDown = cumsum(exp(-r*T)*PayoffSigmaDown)./cumsum(ones(1,size(S,2)));

% Compute running greeks
RunningForwardDelta = (RunningPriceUp - RunningPrice)/DeltaEpsilon;
RunningBackwardDelta = (RunningPrice - RunningPriceDown)/DeltaEpsilon;
RunningDelta = (RunningForwardDelta + RunningBackwardDelta)/2;
RunningGamma = (RunningForwardDelta - RunningBackwardDelta)/(DeltaEpsilon);
RunningForwardVega = (RunningPriceSigmaUp - RunningPrice)/VegaEpsilon;
RunningBackwardVega = (RunningPrice - RunningPriceSigmaDown)/VegaEpsilon;
RunningVega = (RunningForwardVega + RunningBackwardVega)/2;

% Identify greek as end of running series
Delta = RunningDelta(length(RunningDelta));
Gamma = RunningGamma(length(RunningGamma));
Vega = RunningVega(length(RunningVega));

DeltaSNR = abs(Delta/std(RealizationDelta));
GammaSNR = abs(Gamma/std(RealizationGamma));
VegaSNR = abs(Vega/std(RealizationVega));

end
```

File GBMAsianMalliavin.m

```
function [ Delta RunningDelta DeltaSNR Gamma RunningGamma GammaSNR Vega RunningVega VegaSNR ] = ...
   GBMAsianMalliavin( PayoffStructure, S, W, K, r,sigma, T )

% This function uses the malliavin method to determine sensitivities.
% Outputs for each greek
%         1) estimated sensitivities
%         2) running estimates of sensitivities as simulation progresses.
%         3) signal to noise ratios of the estimators.

RealizationVega = 0;

% Initialize
WT = W(size(W,1),:);
ST = S(size(S,1),:);
S0 = S(1,1);
STBar = sum(S,1)/size(S,1);

% The payoffs under each realization in S
if (PayoffStructure == 'B')
        Payoff = STBar > K;
else
```

```
        Payoff = max(STBar - K,0);
end

% The computed Greeks for each realization
RealizationDelta = (exp(-r*T)/(sigma^2*S0))*Payoff.*(2*(ST - S0)./STBar + sigma^2 -2*r);
RealizationGamma = ((4*exp(-r*T)/(sigma^3*S0^2))*Payoff.*((((ST - S0).^2 - (ST - S0)*r.*STBar)./ ...
(sigma*(STBar.^2))) - (sigma*S0)./STBar)) - ((2*r)/(sigma^2*S0))*RealizationDelta;

% The running average Greeks
RunningDelta = cumsum(RealizationDelta)./cumsum(ones(1,size(S,2)));
RunningGamma = cumsum(RealizationGamma)./cumsum(ones(1,size(S,2)));
RunningVega = cumsum(RealizationVega)./cumsum(ones(1,size(S,2)));

% The computed average Greeks
Delta = RunningDelta(length(RunningDelta));
Gamma = RunningGamma(length(RunningGamma));
Vega = RunningVega(length(RunningVega));

DeltaSNR = abs(Delta/std(RealizationDelta));
GammaSNR = abs(Gamma/std(RealizationGamma));
VegaSNR = abs(Vega/std(RealizationVega));


end
```

File GBMAsianPathwise.m

```
function [ Delta RunningDelta DeltaSNR Gamma RunningGamma GammaSNR Vega RunningVega VegaSNR ] = ...
   GBMAsianPathwise ( PayoffStructure, S, W, K, r,sigma, T )

% This function uses the pathwise derivative method to determine sensitivities.
% Outputs for each greek
%         1) estimated sensitivities
%         2) running estimates of sensitivities as simulation progresses.
%         3) signal to noise ratios of the estimators.

% Initialize

RealizationVega = 0;

% Initialize
WT = W(size(W,1),:);
ST = S(size(S,1),:);
S0 = S(1,1);
STBar = sum(S,1)/size(S,1);

% The payoffs under each realization in S
if (PayoffStructure == 'B')
        PayoffDeriv = 0;
        PayoffSecondDeriv = 0;
else
        PayoffDeriv = STBar > K;
        PayoffSecondDeriv = 0;
end

% The computed Greeks for each realization
```

```
RealizationDelta = (exp(-r*T)/S0)*PayoffDeriv.*ST;
RealizationGamma = (exp(-r*T)/S0^2)*PayoffSecondDeriv.*(ST.^2);
RealizationVega = exp(-r*T)*PayoffDeriv.*ST.*(WT - sigma*T);

% The running average Greeks
RunningDelta = cumsum(RealizationDelta)./cumsum(ones(1,size(S,2)));
RunningGamma = cumsum(RealizationGamma)./cumsum(ones(1,size(S,2)));
RunningVega = cumsum(RealizationVega)./cumsum(ones(1,size(S,2)));

% The computed average Greeks
Delta = RunningDelta(length(RunningDelta));
Gamma = RunningGamma(length(RunningGamma));
Vega = RunningVega(length(RunningVega));

DeltaSNR = abs(Delta/std(RealizationDelta));
GammaSNR = abs(Gamma/std(RealizationGamma));
VegaSNR = abs(Vega/std(RealizationVega));

end
```

This script summarizes the data gathered in an easily readable format.

File Printer.m

```
function Printer ( Method, DeltaEstimate, DeltaActual, DeltaSNR, GammaEstimate, GammaActual, ...
   GammaSNR, VegaEstimate, VegaActual, VegaSNR)

HeadString = [Method];
DeltaString = ['\Delta: ' num2str(DeltaEstimate) ' (est.), ' num2str(DeltaActual) ' (act.), error = ' ...
    num2str(abs(100*(DeltaEstimate - DeltaActual)/DeltaActual)) '%, snr = ' num2str(DeltaSNR) ];
GammaString = ['\Gamma: ' num2str(GammaEstimate) ' (est.), ' num2str(GammaActual) ' (act.), error = ' ...
    num2str(abs(100*(GammaEstimate - GammaActual)/GammaActual)) '%, snr = ' num2str(GammaSNR)];
VegaString = ['\Vega: ' num2str(VegaEstimate) ' (est.), ' num2str(VegaActual) ' (act.), error = ' ...
   num2str(abs(100*(VegaEstimate - VegaActual)/VegaActual)) '%, snr = ' num2str(VegaSNR)];
String = [HeadString 10 DeltaString 10 GammaString 10 VegaString 10 ];
disp(String)
```

BIBLIOGRAPHY

[1] E. Benhamou. An application of Malliavin calculus to continuous time asian options greeks. LSE Working Paper. Available at SSRN: http://ssrn.com/abstract=265284 or DOI: 10.2139/ssrn.265284, May 2000.

[2] E. Benhamou. Smart Monte Carlo: Various tricks using Malliavin calculus. Written for Goldman Sachs Fixed Income Strategy group, January 2001.

[3] F.E. Benth. *Option Theory with Stochastic Analysis: An Introduction to Mathematical Finance*. Springer Verlag, 2004.

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

[5] M. Broadie and P. Glasserman. Estimating security price derivatives using simulation. *Management Science*, 42:269–285, 1996.

[6] M.H.A. Davis and M.P. Johansson. Malliavin Monte Carlo greeks for jump diffusions. *Stochastic processes and their applications*, 116(1):101–129, 2006.

[7] F. Delbaen and W. Schachermayer. A general version of the fundamental theorem of asset pricing. *Mathematische Annalen*, 300(1):463–520, 1994.

[8] E. Fournié, J.-M. Lasry, J. Lebuchoux, and P.-L. Lions. Applications of Malliavin calculus to Monte Carlo methods in finance II. *Finance and Stochastics*, 5:201–236, 2001.

[9] E. Fournié, J.-M. Lasry, J. Lebuchoux, P.-L. Lions, and N. Touzi. Applications of Malliavin calculus to Monte Carlo methods in finance. *Finance and Stochastics*, 3:391–412, 1999.

[10] P. Friz. An introduction to Malliavin calculus. Retrieved from `http://www.statslab.cam.ac.uk/~peter/malliavin/Malliavin2005/mall.pdf`, 2005.

[11] T. Gard. *Introduction to Stochastic Differential Equations*. Marcel Dekker, 1988.

[12] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2003.

[13] S.L. Heston. A closed–form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2):327–343, 1993.

[14] Y.C. Ho and X.-R. Cao. Optimization and perturbation analysis of queuing networks. *Journal of Optimization Theory and Applications*, 40:559–582, 1983.

[15] J.L. Howland and R. Vaillancourt. A generalized curve-fitting procedure. *Journal of the Society for Industrial and Applied Mathematics*, 9(2):165–168, 1961.

[16] A. Kohatsu-Higa and M. Montero. Malliavin calculus in finance. In S. Rachev, editor, *Handbook of computational and numerical methods in finance*, pages 111–174. Birkhauser Verlag, Boston, 2004.

[17] P. Malliavin. Stochastic calculus of variation and hypoelliptic operators. In K. Itô, editor, *Proceedings of the International Symposium on Stochastic Differential Equations*, pages 195–263. Res. Inst. Math. Sci., Kyoto Univ., 1976.

[18] M. Mrad, N. Touzi, and A. Zeghal. Monte Carlo estimation of a joint density using Malliavin calculus, and application to American options. *Computational Economics*, 27(4):497–531, 2006.

[19] S.N. Neftci. *An Introduction to the Mathematics of Financial Derivatives.* Academic Press, 2nd edition, 1999.

[20] D. Nualart. *The Malliavin Calculus and Applications.* Springer, 1995.

[21] B. Øksendal. An introduction to Malliavin calculus with applications to economics. Unpublished notes from the Norwegian School of Economics and Business Administration's course BF 05: Malliavin Calculus with Applications to Economics, May 1997.

[22] S. Ross. *Introduction to Probability Models.* Academic Press, 9th edition, 2007.

[23] H.L. Royden. *Real Analysis.* Prentice Hall, 3rd edition, 1988.

[24] T. Schröter. Malliavin calculus in finance. Unpublished special topic essay from University of Oxford, Trinity College in 2007., 2007.

[25] R. Suri and M. Zazanis. Perturbation analysis gives strongly consistent sensitivity estimates for the M/G/1 queue. *Management Science*, 34:39–64, 1988.