Estimating expected first passage times using multilevel Monte Carlo algorithm

Tom Primožič

New College

University of Oxford

MSc in Mathematical and Computational Finance

June 2011

Mojim staršem, za vse, kar sta mi dala,
mojemu bratu, ki me že celo življenje prenaša,
in mojim prijateljem, ki me sprejemajo takšnega, kot sem.

Pomenite mi vse.

To my parents, for everything they have given me,
to my brother, who put up with me his whole life,
and to my friends, who accept me for who I am.

You mean the world to me.

Acknowledgements

I would like to thank Mike Giles for his help and guidance, and for his seemingly unending patience.

Abstract

In this paper we devise a method of numerically estimating the expected first passage times of stochastic processes. We use Monte Carlo path simulations with Milstein discretisation scheme to approximate the solutions of scalar stochastic differential equations. To further reduce the variance of the estimated expected stopping time and improve computational efficiency, we use the multi-level Monte Carlo algorithm, recently developed by Giles (2008a), and other variance-reduction techniques. Our numerical results show significant improvements over conventional Monte Carlo techniques.

Contents

1 Introduction

In this work, we are interested in developing efficient numerical methods for estimating expected first passage times of stochastic processes. The *first passage time*, also known as the *first hitting time* or the *first exit time*, is the first time that a given stochastic process reaches a certain level, the *barrier*. First hitting times have many applications in finance, the theory of investment decisions, and engineering.

Analytic, closed-form expressions for the expected first passage times are known only for some very basic cases, such as arithmetic and geometric Brownian motion with a static barrier (we derive these formulas in section 2). The aim of this work is to develop an efficient and accurate numerical method for estimating expected first passage times for the case of a general continuous stochastic process, driven by the stochastic differential equation (SDE)

$$dX_0 = a(X_t, t)dt + b(X_t, t)dW_t. \tag{1}$$

The first passage time of the process X_t with barrier B is defined as

$$\tau = \inf \{u \mid X_u = B\}. \tag{2}$$

We use the theory developed in section 2 to check the accuracy of our algorithms in the simple cases.

There are two basic methods of approximating functions of solutions of SDEs: the finite difference method and the Monte Carlo algorithm. Although finite differences produce very accurate results in simple cases, they are very computationally expensive for higher-dimensional problems. On the other hand, although less accurate, Monte Carlo algorithms are relatively simple to implement and extend to more complex cases, and allow for an efficient parallelization of the algorithm using multiple CPUs or GPUs.

However, naïve implementations of Monte Carlo simulation turn out to be rather inefficient for estimating expected first passage times. The simplest scheme for approximating the paths of a process driven by (1) is the Euler-Maruyama method, the stochastic extension of the Euler method for approximating solutions to ordinary differential equations. Under certain conditions on the coefficients of (1), the Euler-Maruyama method has a first order weak convergence for payoffs of European options, which is reduced to a half order weak convergence for some cases of path-dependent payoffs, such as the first stopping times (Higham et al., 2011). To improve the order of weak convergence, we use Brownian interpolation to calculate the probability of hitting the barrier (Glasserman, 2004), and utilize the multilevel Monte

Carlo algorithm, first developed by Giles (2008a). Furthermore, as the first passage times depend heavily on the actual path taken by the process, the other important metric is the order of strong convergence, which is one half for the Euler-Maruyama method. Therefore, in contrast to the approach taken by Higham et al. (2011) in their adaptation of the multilevel Monte Carlo method, we use the Milstein scheme to discretely approximate the simulated paths, thus achieving an order one strong convergence.

Expected stopping times have uses in physics (Jia and Li, 1996; Gitterman, 2000; Bedingham, 2008), real options analysis (Dias, Appendix A), and investing (Trabelsi, 2007). Furthermore, our algorithm is extensible to a wide variety of other related problems. It could be used to estimate the expected value of functions of τ, such as the expected discount factor $\mathbb{E}\left[e^{-r\tau}\right]$, which can be useful in pricing parisian options (Gauthier, 2002), barrier options with rebates, and American digital options. The algorithm is also easily adaptable to other models, such as SDEs driven by jump-diffusions (Atiya and Metwally, 2005) and models with time-dependent barriers (Lo et al., 2009).

The rest of this work is organized as follows. In section 2 we derive some theoretical properties of the Brownian bridge process and of the first passage times for arithmetic and geometric Brownian motion. In section 3 we briefly describe the standard Monte Carlo algorithm and its asymptotic computational complexity. In section 4 we present the multilevel Monte Carlo algorithm and explain how it achieves a better computational efficiency than standard Monte Carlo. We also describe in detail the algorithms used in approximating the (expected) first passage time of a simulated path. In section 5 we compare the numerical results of different algorithms. Section 6 concludes.

2 First passage time

In this section, we look at some theoretical properties of Brownian motion. First, we state the distribution of the first passage time of the drifted Brownian motion, and derive the distributions for the arithmetic Brownian motion and for the geometric Brownian motion. Then, we find the distribution of the minimum of the Brownian bridge, that is, a (drifted) Brownian motion conditioned on the value at the end point of the interval.

We begin with a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let W_t be a standard $\mathbb{P}$-Brownian motion and let $W_t^\alpha = \alpha t + W_t$ denote a drifted Brownian motion with drift α. Without

loss of generality we will only consider its running minimum M_t^α:

$$M_t^\alpha = \min_{0 \le u \le t} W_u^\alpha = \min_{0 \le u \le t} \{\alpha u + W_u\}$$

Using the change of measure, Girsanov theorem and the Reflection principle of the Brownian motion, we can get the joint distribution of (M_t^α, W_t^α), which we can then use to derive both the unconditional distribution of the minimum, and the conditional distribution of minimum M_t^α given W_t^α.

The distribution function of the minimum of the drifted Brownian motion is

$$\mathbb{P}\left[M_t^\alpha \le m\right] = \Phi\left(\frac{m - \alpha t}{\sqrt{t}}\right) + e^{2m\alpha}\Phi\left(\frac{m + \alpha t}{\sqrt{t}}\right),$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. Let $\tau_B^\alpha = \inf\{u \mid W_u^\alpha \le B\}$ be the first time that the drifted Brownian motion W_t^α touches the barrier $B < 0$, i.e. the first passage time (the process never hitting the barrier corresponds to infimum of an empty set, which is ∞). The distributions of the first passage time and the minimum are strongly connected; indeed, the event $\{\tau_B^\alpha > t\}$ is the same as the event $\{M_t^\alpha > B\}$. We can thus write down the distribution of τ_B^α:

$$\begin{aligned}\mathbb{P}\left[\tau_B^\alpha \le t\right] &= \mathbb{P}\left[M_t^\alpha \le B\right] \\ &= \Phi\left(\frac{B - \alpha t}{\sqrt{t}}\right) + e^{2B\alpha}\Phi\left(\frac{B + \alpha t}{\sqrt{t}}\right). \end{aligned} \tag{3}$$

If $\alpha > 0$ and the process is drifting away from the barrier, we can see that when $t \to \infty$, the probability of W_t^α hitting the barrier is $\mathbb{P}\left[\tau_B^\alpha < \infty\right] = e^{2B\alpha} < 1$, which means that there is a nonzero probability that the process never reaches the barrier and therefore the expected fist passage time is ∞. Similarly, if $\alpha = 0$, the distribution of the first passage time is the Lévy distribution with location parameter $\gamma = 0$ and scale $\sigma = B^2$, which is continuous, but nonetheless has infinite expectation.

Finally, if $\alpha < 0$ and the process drifts towards the barrier, the first passage time τ_B^α is distributed according to the inverse Gaussian distribution with mean $\mu = \frac{B}{\alpha}$ and shape parameter $\lambda = B^2$.

$$\tau_B^\alpha \sim IG\left(\frac{B}{\alpha}, B^2\right).$$

Let us now consider the case of an arithmetic Brownian motion following the SDE $dX_t = adt + bdW_t$ with the initial value X_0. Let $Y_t = \min_{0 \le u \le t} X_t$ denote its running

minimum and $\tau = \inf\{u \mid X_u \leq B\}$ the first passage time for the barrier B. Assume, as above, that $\alpha < 0$ and that $B - X_0 < 0$. We see that the solution to the SDE is

$$X_t = X_0 + at + bW_t = X_0 + b\left(\frac{a}{b}t + W_t\right) = X_0 + bW_t^{\frac{a}{b}}$$

and that we can write

$$Y_t = \min_{0\leq u\leq t} X_u = \min_{0\leq u\leq t}\left\{X_0 + bW_u^{\frac{a}{b}}\right\} = X_0 + bM_t^{\frac{a}{b}}.$$

Using these equations, we can express τ in terms of the first passage time of the drifted Brownian motion:

$$\begin{aligned}\mathbb{P}[\tau \leq t] = \mathbb{P}[Y_t \leq B] = \mathbb{P}[X_0 + bM_t^{\frac{a}{b}} \leq B] = \\ = \mathbb{P}\left[M_t^{\frac{a}{b}} \leq \frac{B - X_0}{b}\right] = \mathbb{P}\left[\tau_{\frac{B-X_0}{b}}^{\frac{a}{b}} \leq t\right]\end{aligned}$$

We see that the first passage time again has the inverse Gaussian distribution, $\tau \sim IG\left(\frac{B-X_0}{a}, \frac{(B-X_0)^2}{b^2}\right)$, and that $\mathbb{E}[\tau] = \frac{B-X_0}{a}$. The expected value does not depend on the diffusion coefficient of (X_t), and can be thought of as *the distance to be traveled* $B - X_0$ divided by *the average speed* a. Indeed, when $b \to 0$ and the stochastic part of the process (X_t) disappears, τ is no longer random, and must be equal to $\frac{B-X_0}{a}$.

Consider now the case of geometric Brownian motion, which is a solution to the SDE $dS_t = \mu S_t dt + \sigma S_t dW_t$ with the initial value S_0. We are interested in the expected first passage time $\rho = \inf\{u \mid S_u \leq L\}$ for the barrier L. The solution to the above SDE is

$$S_t = S_0 \exp\left(\sigma W_t + \left(\mu - \frac{1}{2}\sigma^2\right)t\right) = \exp\left(\log S_t + \left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t\right).$$

Substituting $S_t = e^{X_t}$ into ρ, we get

$$\rho = \inf\{u \mid S_u \leq L\} = \inf\{u \mid e^{X_u} \leq L\} = \inf\{u \mid X_u \leq \log L\}.$$

We can therefore see that this case is equivalent to the case with arithmetic Brownian motion (X_t) with parameters $X_0 = \log S_0$, $a = \mu - \frac{1}{2}\sigma^2$, $b = \sigma$, and barrier $B = \log L$. If the conditions $B - X_0 < 0 \iff L < S_0$ and $a < 0 \iff \mu < \frac{1}{2}\sigma^2$ are satisfied, the first passage time ρ is distributed as

$$\rho \sim IG\left(\frac{\log L - \log S_0}{\mu - \frac{1}{2}\sigma^2}, \frac{(\log L - \log S_0)^2}{\sigma^2}\right). \tag{4}$$

We return now back to the case of drifted Brownian motion. Using the joint distribution of (M_t^α, W_t^α) and the marginal distribution of W_t^α, we can find the conditional probability distribution of M_t^α given W_t^α:

$$\mathbb{P}\left[M_t^\alpha = m \mid W_t^\alpha = w\right] = e^{\frac{2m(w-m)}{t}} = \exp\left(-\frac{(w-2m)^2}{2t} + \frac{w^2}{2t}\right)$$

This is the distribution of the minimum of a Brownian bridge on the interval $[0,t]$ with endpoint values $W_0^\alpha = 0$ and $W_t^\alpha = w$. We can see that the distribution is independent of the drift coefficient α; knowing the end values also determines the expected drift of the process between those end values.

If we know the probability function $F(z)$ of a continuous random variable Z, we can use a uniformly distributed random variable to generate variables distributed as Z. Define the generalized inverse $F^{\leftarrow}$ as follows:

$$F^{\leftarrow}(u) = \inf\left\{z \mid F(z) = u\right\}, 0 \le u \le 1$$

If $U \sim U([0,1])$ is random variable uniformly distributed on the interval $[0,1]$, then $F^{\leftarrow}(U) \overset{(d)}{=} Z$, where $\overset{(d)}{=}$ denotes equality in distribution (Devroye, 1986, Theorem 2.1). We will use this in our Monte Carlo simulation to generate the minimum value of a Brownian bridge. Conditioning M_t^α on $W_t^\alpha = w$, we have

$$M_t^\alpha \overset{(d)}{=} \frac{1}{2}\left(w - \sqrt{w^2 - 2t\log U}\right).$$

Finally, let us return to the arithmetic Brownian motion (X_t). Since the process (X_t) has stationary and independent increments, we can write

$$X_t \overset{(d)}{=} X_s + bW_{\Delta t}^{\frac{a}{b}},$$

where $\Delta t = t - s$. Then the minimum $Y_{s,t}$ of the process (X_t) on the interval $[s,t]$ is distributed as

$$\begin{aligned}
\mathbb{P}\left[Y_{s,t} \le y \mid X_s, X_t\right] &= \mathbb{P}\left[X_s + bM_{\Delta t}^{\frac{a}{b}} \le y \mid X_s + bW_{\Delta t}^{\frac{a}{b}} = X_t\right] \\
&= \mathbb{P}\left[M_{\Delta t}^{\frac{a}{b}} \le \frac{y - X_s}{b} \mid W_{\Delta t}^{\frac{a}{b}} = \frac{X_t - X_s}{b}\right] \\
&= \exp\left(-\frac{2(X_s - y)(X_t - y)}{b^2 \Delta t}\right).
\end{aligned}$$

If $U \sim U([0,1])$, then we can generate $Y_{s,t}$ as follows:

$$\begin{aligned}
Y_{s,t} &\overset{(d)}{=} X_s + bM_{\Delta t}^{\frac{a}{b}} \\
&\overset{(d)}{=} X_s + \frac{1}{2}b\left(\frac{X_t - X_s}{b} - \sqrt{\left(\frac{X_t - X_s}{b}\right)^2 - 2\Delta t \log U}\right) \\
&= \frac{1}{2}\left(X_t + X_s - \sqrt{(X_t - X_s)^2 - 2b^2\Delta t \log U}\right) \qquad (5)
\end{aligned}$$

We will use this formula in section 4.2.

3 Monte Carlo

In this section, we present a brief overview of the Monte Carlo technique of estimating quantities using randomness. We begin by explaining basic terms and defining notation. We then introduce the concepts weak and strong convergence and describe the Euler-Mayurama and Milstein discretisation schemes. In the end, we focus on accuracy and computational complexity.

Monte Carlo is a method of numerically estimating some quantity which can be represented as the expected value $\mathbb{E}[Y]$, where $Y = f(X)$ is a functional of a random variable X with an infinite sample space. The most basic estimate of $\mathbb{E}[Y]$ is

$$\widehat{Y} = \frac{1}{N} \sum_{i=1}^{N} \widehat{f}\left(\widehat{X}^{(i)}\right),$$

where $\widehat{X}^{(i)}$ is a random outcome of a simulation that discretely approximates X, $\widehat{f}$ is a discrete approximation to f, and N is the number of independent simulations of X. Then, it is trivial to see that $\mathbb{E}[\widehat{Y}] = \mathbb{E}\left[\widehat{f}(\widehat{X})\right]$.

We measure the error of a single run of Monte Carlo with $e = \widehat{Y} - \mathbb{E}[f(X)]$, which is the difference between the estimate we got and the estimate we have wanted. The accuracy of a particular algorithm can be measured by the expected absolute error, or more often by the *root mean square error* $\sqrt{\mathbb{E}[e^2]}$. Our goal in the design of Monte Carlo algorithms is to estimate Y with accuracy ε as efficiently as possible, that is, to minimize the computational complexity required to achieve the root mean square error of ε.

We can express the mean square error as

$$\begin{aligned}
\mathbb{E}\left[e^2\right] &= \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[f(X)]\right)^2\right] \\
&= \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[\widehat{Y}] + \mathbb{E}[\widehat{Y}] - \mathbb{E}[f(X)]\right)^2\right] \\
&= \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[\widehat{Y}]\right)^2\right] + \left(\mathbb{E}\left[\widehat{f}(\widehat{X})\right] - \mathbb{E}[f(X)]\right)^2 \qquad (6)
\end{aligned}$$

The first term in this expression is the variance of the estimator

$$\mathrm{Var}[\widehat{Y}] = \frac{1}{N^2} \mathrm{Var}\left[\sum_{i=1}^{N} \widehat{f}(\widehat{X})\right] = \frac{1}{N} \mathrm{Var}\left[\widehat{f}(\widehat{X})\right]$$

and is proportional to $\frac{1}{N}$, and the second term is the square of the *bias* of the approximation $\widehat{f}(\widehat{X})$.

In our applications, the random quantity X is a stochastic process (X_t) that is driven by the SDE (1). Even a single realization, or *path*, of the process (X_t) is ∞-dimensional and thus cannot be accurately represented in a computer. That is why we use small positive timesteps $h = \Delta t$ to approximate the infinitesimal timesteps dt, and limit our simulation to a finite interval $[0, T]$. We will use $\widehat{X}_i$ as an approximation of X_{t_i}, where $t_i = i\Delta t$, $i = 0, \ldots, K$. We see that $K = \frac{T}{h}$ is the number of timesteps in the interval $[0, T]$. We generate the simulated path starting with $\widehat{X}_0 = X_0$ and at each timestep i calculate the value of the process at the next timestep, $\widehat{X}_{i+1}$, using the increments of the underlying stochastic process ΔW_i.

Ideally, we want the *discretisation error* $\widehat{f}(\widehat{X}) - f(X)$ to go to 0 as $h \to 0$ and $K \to \infty$, i.e. that the approximation converges. There are two ways of defining the convergence of the approximated path. The first is the notion of *strong convergence*, which means that the absolute distance between the paths X and $\widehat{X}$ goes to zero as $h \to 0$. We say that the order of strong convergence is m if

$$\mathbb{E}\left[\sup_{0 \le i \le K} \left|\widehat{X}_i - X_{t_i}\right|\right] = O(h^m).$$

The other measure of convergence is *weak convergence*, which refers to the convergence of the difference between the expected estimated value and the expected actual value. The order of weak convergence is m if

$$\left|\mathbb{E}\left[\widehat{f}(\widehat{X}) - f(X)\right]\right| = O(h^m).$$

There are multiple ways of calculating the values $\widehat{X}_i$, and different methods offer different tradeoffs. The most basic one is the Euler-Maruyama scheme, a stochastic generalization of the Euler method for numerically approximating ordinary differential equations:

$$\widehat{X}_{i+1} = \widehat{X}_i + a_i h + b_i \Delta W_i,$$

where $a_i = a(\widehat{X}_i, t_i)$ and $b_i = b(\widehat{X}_i, t_i)$ are the coefficients of the SDE and $\Delta W_i = W_{i+1} - W_i \sim N(0, \sqrt{\Delta t})$ is the increment of the simulated Brownian motion. Under certain assumptions about the coefficients $a(X, t)$ and $b(X, t)$, the Euler-Maruyama method has strong convergence of order $\frac{1}{2}$ and weak convergence of order 1 for smooth functions f of the final value X_t (Kloeden and Platen, 1992). Another popular discretisation scheme is the Milstein scheme:

$$\widehat{X}_{i+1} = \widehat{X}_i + a_i h + b_i \Delta W_i + \frac{1}{2} b_i \frac{\partial b}{\partial X}(X_i, t_i)\left(\Delta W_i^2 - h\right).$$

The Milstein scheme requires the calculation of another term, which involves the partial derivative of b with respect to X, but in return provides order 1 weak convergence and an improved $O(h)$ strong convergence (Kloeden and Platen, 1992).

Returning to the mean square error (6), we see that the second term, the square of the bias of the estimator, is related to the weak convergence of the discretisation method used. As the variance of the estimator is proportional to $\frac{1}{N}$ and, assuming order 1 weak convergence, square of bias is proportional to h^2, we see that mean square error is

$$\mathbb{E}\left[e^2\right] = O\left(\frac{1}{N}\right) + O\left(h^2\right).$$

To ensure that the root mean square error is proportional to ε, we must have $\mathbb{E}\left[e^2\right] = O(\varepsilon^2)$ and therefore $\frac{1}{N} = O\left(\varepsilon^2\right)$ and $h^2 = O\left(\varepsilon^2\right)$, which means that $N = O\left(\varepsilon^{-2}\right)$ and $h = O(\varepsilon)$. On the other hand, the computational cost of the algorithm is proportional to the multiple of N, the number of simulated sample paths, and $K = \frac{T}{h}$, the number of timesteps in each sample path. Therefore, the cost is $C = O(NK) = O\left(Nh^{-1}\right) = O\left(\varepsilon^{-3}\right)$. That means that the computational complexity, required to achieve a given accuracy ε using the Monte Carlo algorithm, is proportional to ε^{-3}.

4 Multi-level Monte Carlo

In this section, we present the multilevel Monte Carlo algorithm and explain the improvements in performance that it offers. We then describe the precise multilevel Monte Carlo algorithm followed by our program. Last, we list 3 different estimators that are used in our numerical calculations, and show how performance can be improved even further.

The multilevel Monte Carlo method was first developed by Giles (2008a) and includes estimating the same value on different *levels* l, using a different timestep h_l for each level. Each next level has M times more timesteps than the previous one, where $M = 2$ or $M = 4$, so $h_l = M^{-l}T$. As in the standard Monte Carlo, we are again estimating $\mathbb{E}[Y] = \mathbb{E}[f(X)]$ using the estimate $\widehat{Y}$, which in the multilevel Monte Carlo algorithm is the sum of estimates for different levels:

$$\widehat{Y} = \sum_{l=0}^{L} \widehat{Y}_l,$$

where

$$\widehat{Y}_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} \widehat{P}_0^{(i)}$$

and

$$\widehat{Y}_l = \frac{1}{N_l} \sum_{i=1}^{N_l} \left(\widehat{P}_l^{(i)} - \widehat{P}_{l-1}^{(i)} \right).$$

Here, $\widehat{P}_l$ denotes the approximation $\widehat{f}(\widehat{X})$ of the value $f(X)$ for a single sample path, simulated using N_l timesteps of size h_l. As indicated by the subscripts $^{(i)}$, the approximations $\widehat{P}_l$ and $\widehat{P}_{l-1}$ are computed from the same sample path, using the same Brownian increments. This will later aid us in reducing the variance of the estimator $\widehat{Y}$. Also note that we are simulating N_l paths on each level, and each sample path is simulated using $\frac{T}{h_L}$ timesteps, so the overall computational cost is proportional to $\sum_{l=0}^{L} N_l h_l^{-1}$.

We can quickly see that

$$\mathbb{E}[\widehat{P}_L] = \mathbb{E}[\widehat{P}_0] + \sum_{l=1}^{L} \left[\widehat{P}_l - \widehat{P}_{l-1} \right]$$

and that for $l = 1, \ldots, L$

$$\mathbb{E}[\widehat{Y}_l] = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathbb{E}\left[\widehat{P}_l - \widehat{P}_{l-1} \right] = \mathbb{E}\left[\widehat{P}_l - \widehat{P}_{l-1} \right],$$

therefore

$$\mathbb{E}[\widehat{Y}] = \sum_{l=0}^{L} \mathbb{E}[\widehat{Y}_l] = \mathbb{E}[\widehat{P}_0] + \sum_{l=1}^{L} \mathbb{E}\left[\widehat{P}_l - \widehat{P}_{l-1} \right] = \mathbb{E}[\widehat{P}_L].$$

Therefore, although we are using different levels with different discretisation errors to estimate $\mathbb{E}[f(X)]$, the final accuracy depends on the accuracy of the finest level L.

Again, to minimize the computational costs necessary for a given accuracy ε, we analyze and minimize the mean square error of $\widehat{Y}$.

$$\begin{aligned} \mathbb{E}[e^2] &= \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}\left[f(X) \right] \right)^2 \right] \\ &= \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[\widehat{Y}] \right)^2 \right] + \left(\mathbb{E}[\widehat{P}_L] - \mathbb{E}\left[f(X) \right] \right)^2 \end{aligned} \tag{7}$$

The first term is the variance:

$$\mathrm{Var}[\widehat{Y}] = \sum_{l=0}^{L} \mathrm{Var}[\widehat{Y}_l] = \sum_{l=0}^{L} \frac{1}{N_l} V_l,$$

where $V_l = \mathrm{Var}\left[\widehat{P}_l - \widehat{P}_{l-1} \right]$. To minimize the variance of $\widehat{Y}$ for a fixed computational cost $C = \sum_{l=0}^{L} N_l h_l^{-1}$, we can treat N_l as continuous variables and use the Lagrange function to find the minimum:

$$\mathcal{L} = \sum_{l=0}^{L} \frac{1}{N_l} V_l + \lambda \left(\sum_{l=0}^{L} N_l h_l^{-1} - C \right)$$

The first order conditions are

$$\frac{\partial \mathcal{L}}{\partial N_l} = -\frac{1}{N_l^2} V_l + \lambda h_l^{-1} = 0,$$

therefore

$$N_l = \lambda^{-\frac{1}{2}} \sqrt{h_l V_l}. \tag{8}$$

Assuming order $\frac{1}{2}$ strong convergence and that the function f is Lipschitz, the variance $\mathrm{Var}\left[\widehat{P}_l - \widehat{P}_{l-1}\right] = O(h_l)$ (Giles, 2008a), so the optimal N_l is proportional to h_l. If we set $N_l = O\left(\varepsilon^{-2} L h_l\right)$, then the overall variance is proportional to ε^2:

$$\mathrm{Var}[\widehat{Y}] = \sum_{l=0}^{L} \frac{V_l}{N_l} = \sum_{l=0}^{L} O(\varepsilon^2 L^{-1} h_l^{-1}) O(h_l) = \sum_{l=0}^{L} \frac{O(\varepsilon^2)}{L} = O(\varepsilon^2).$$

The second term of (7) is the square of the bias due to weak error. Assuming $O(h_l)$ weak convergence for an individual level l, we can see that the bias of the overall method is $O(h_L) = O(M^{-L})$. To achieve the desired accuracy, we want the bias to be proportional to ε, so we set

$$L = -\log_M \varepsilon = \frac{\log \varepsilon^{-1}}{\log M} = O(\log \varepsilon^{-1}).$$

Summarizing the above estimates about the asymptotic behavior of the multilevel Monte Carlo algorithm, we can see that the overall computational complexity is proportional to

$$\begin{aligned} \sum_{l=0}^{L} N_l h_l^{-1} &= \sum_{l=0}^{L} O(\varepsilon^{-2} L h_l) h_l^{-1} \\ &= O(\varepsilon^{-2} L^2) = O(\varepsilon^{-2} (\log \varepsilon)^2). \end{aligned}$$

It is easy to see that $\varepsilon^{-1} \gg (\log \varepsilon)^2$ as $\varepsilon \to 0$. We can therefore see that the multilevel Monte Carlo method has a significantly better asymptotic computational complexity required to achieve a given accuracy level than the standard Monte Carlo method.

The above analysis can be summarized with the following theorem.

Theorem 4.0.1 (Complexity theorem (Giles, 2008a, Theorem 3.1)**)**
Let P denote a functional of the solution of stochastic differential equation (1) *for a given Brownian path W_t, and let $\widehat{P}_l$ denote the corresponding approximation using a numerical discretization with timestep $h_l = M^{-l} T$.*

If there exist independent estimators $\widehat{Y}_l$ based on N_l Monte Carlo samples, and positive constants $\alpha \geq \frac{1}{2}$, β, c_1, c_2, c_3 such that

i) $\mathbb{E}[\widehat{P}_l - P] \leq c_1 h_l^\alpha$

ii) $\mathbb{E}[\widehat{Y}_l] = \begin{cases} \mathbb{E}[\widehat{P}_0], & l = 0 \\ \mathbb{E}[\widehat{P}_l - \widehat{P}_{l-1}], & l > 0 \end{cases}$

iii) $\mathrm{Var}[\widehat{Y}_l] \leq c_2 N_l^{-1} h_l^\beta$

iv) *C_l, the computational complexity of $\widehat{Y}_l$, is bounded by*

$$C_l \leq c_3 N_l h_l^{-1},$$

then there exists a positive constant c_4 such that for any $\varepsilon < e^{-1}$ there are values L and N_l for which the multilevel estimator

$$\widehat{Y} = \sum_{l=0}^{L} \widehat{Y}_l$$

had a mean-square-error with bound

$$MSE := \mathbb{E}\left[\left(\widehat{Y} - \mathbb{E}[P]\right)^2\right] < \varepsilon^2$$

with a computational complexity C with bound

$$C \leq \begin{cases} c_4 \varepsilon^{-2}, & \beta > 1 \\ c_4 \varepsilon^{-2} (\log \varepsilon)^2, & \beta = 1 \\ c_4 \varepsilon^{-2-(1-\beta)/\alpha}, & 0 < \beta < 1. \end{cases}$$

The theorem tells us that if $\beta > 1$, that is, if the variance of the estimators $\mathrm{Var}[\widehat{Y}_l]$ is decreasing faster than $O(h_l)$ as $h_l \to 0$, then we can further reduce the computational complexity to $O(\varepsilon^{-2})$.

4.1 Multilevel Monte Carlo algorithm

Let us now outline the multilevel Monte Carlo algorithm as executed by a numerical simulation. We begin with $L = 0$ and estimate V_L using an initial $N_L = 100$ samples. We then determine the optimal N_l for $l = 0, \ldots, L$ and generate additional samples as needed. Next, we estimate the accuracy of the method and check whether we have already converged. If we have not, we increase L by 1 and repeat the procedure.

To achieve the goal of root mean square error below ε, we must ensure that the estimator variance $\mathrm{Var}[\widehat{Y}] < \frac{1}{2}\varepsilon^2$ and that the bias $\mathbb{E}\left[\widehat{P}_L - f(X)\right] < \frac{\varepsilon}{\sqrt{2}}$. We control the first by using the optimal number of samples on each level, and the second by increasing the number of levels and thus decreasing the finest timestep.

From (8) we see that $N_l = \lambda^{-\frac{1}{2}}\sqrt{h_l V_l} = \sqrt{h_l V_l}/\sqrt{\lambda}$ and therefore

$$\mathrm{Var}[\widehat{Y}] = \sum_{l=0}^{L} \frac{V_l}{N_l} = \sum_{l=0}^{L} \frac{\sqrt{\lambda}}{\sqrt{h_l V_l}} V_l \leq \frac{1}{2}\varepsilon^2.$$

It is easy to see that

$$\lambda \geq 2\varepsilon^{-2} \sum_{l=0}^{L} \sqrt{V_l/h_l},$$

and thus the optimal number of samples for level l is

$$N_l = \left\lceil 2\varepsilon^{-2}\sqrt{h_l V_l} \left(\sum_{l=0}^{L} \sqrt{V_l/h_l} \right) \right\rceil. \tag{9}$$

We can ensure that the bias $\mathbb{E}\left[\widehat{P}_L - f(X)\right] < \frac{\varepsilon}{\sqrt{2}}$ by using small enough timestep h_L. If we assume order one or better weak convergence, then $\mathbb{E}\left[\widehat{P}_L - f(X)\right] = O(h_L)$ and as $l \to \infty$, we can approximate the remaining bias with a linear function:

$$\mathbb{E}\left[f(X) - \widehat{P}_l\right] = \alpha h_l.$$

Hence

$$\begin{aligned}\mathbb{E}\left[\widehat{P}_L - \widehat{P}_{L-1}\right] &= \mathbb{E}\left[\widehat{P}_L - f(X) + f(X) - \widehat{P}_{L-1}\right] \\ &= \alpha(M-1)h_L = (M-1)\mathbb{E}\left[f(X) - \widehat{P}_L\right]\end{aligned}$$

and we can estimate the remaining bias with $\widehat{Y}_L = \frac{1}{N_L}\sum_{i=1}^{N_L}\left(\widehat{P}_L^{(i)} - \widehat{P}_{L-1}^{(i)}\right)$, which is an estimator for $\mathbb{E}\left[\widehat{P}_L - \widehat{P}_{L-1}\right]$. Therefore, the basic test for convergence is

$$\left|\widehat{Y}_L\right| \leq \frac{M-1}{\sqrt{2}}\varepsilon.$$

However, if we think of $\mathbb{E}\left[\widehat{P}_L - \widehat{P}_{L-1}\right]$ as a function of h_L that converges to zero as $h_L \to 0$, and of $\widehat{Y}_L$ as a discrete sample of the value of this function, we see that the function might change signs before the desired accuracy is reached. If we sample the function at point h_L near its root, we will get $\widehat{Y}_L$ that will be misleadingly small. To avoid this possibility, Giles (2008a) suggests estimating the remaining bias with the estimates of the two finest timesteps, therefore using the following criterion for convergence instead:

$$\max\left\{\frac{1}{M}\left|\widehat{Y}_{L-1}\right|, \left|\widehat{Y}_L\right|\right\} \leq \frac{M-1}{\sqrt{2}}\varepsilon. \tag{10}$$

We can now outline the complete multilevel Monte Carlo algorithm:

1) begin with $L = 0$

2) calculate the initial estimate of V_L using 100 samples

3) determine optimal N_l using (9)

4) generate additional samples for $l = 0, \ldots, L$

5) test for convergence using (10)

6) if we have not converged, set $L := L + 1$ and continue from step 2

4.2 Expected first passage times and multilevel Monte Carlo

We return now back to the aim of this work, describing an algorithm for efficient estimation of expected first passage times of general stochastic processes. Denote by (X_t) the stochastic process, driven by the SDE (1), and by $(\widehat{X}_i)$ its discrete approximation, simulated on K timesteps of size $h = \frac{T}{K}$ by the Milstein discretisation scheme:

$$\widehat{X}_{i+1} = \widehat{X}_i + a_i h + b_i \Delta W_i,$$

where $a_i = a(\widehat{X}_i, t_i)$, $b_i = b(\widehat{X}_i, t_i)$. We are estimating the expected first passage time $\mathbb{E}[\tau]$ on a finite time interval $[0, T]$, therefore $\tau = \tau(X) = \inf\{u \mid X_u \leq B\} \wedge T$, where $B < X_0$ is the specified barrier. We want to approximate τ with $\widehat{\tau} = \widehat{\tau}(\widehat{X})$. However, we only know the approximate value of the simulated process (X_t) at discrete points in time t_i. Below, we outline three different techniques for approximating $\widehat{\tau}$.

The first approximation is based on a straightforward linear interpolation of the process between the values at discrete timesteps. If there exists such i that $\widehat{X}_i \leq B$, we take the smallest such i and assume that the process (X_t) first passed the barrier somewhere in the time interval $[t_{i-1}, t_i]$. Selecting the lower bound of the time interval, we would tend to underestimate the actual stopping time, while using the upper bound would cause us to consistently overestimate, so we choose the middle value $\widehat{\tau} = \frac{1}{2}(t_{i-1} + t_i)$. If $\widehat{X}_i > B$ for all $i = 1, \ldots, \frac{T}{h}$, then we set $\widehat{\tau} = T$. We call this method the *simple* method.

$$\widehat{\tau}_s = \begin{cases} \frac{1}{2}(t_{i-1} + t_i), & \text{if } i = \min\left\{k \mid \widehat{X}_k \leq B\right\} \text{ exists} \\ T & \text{otherwise} \end{cases}$$

We can improve the simple approximation by noting that the the barrier B can be reached between two timesteps t_i and t_{i+1} even if both $\widehat{X}_i > B$ and $\widehat{X}_{i+1} > B$. If we assume that the coefficients $a(X_t, t)$ and $b(X_t, t)$ are constant over the whole

interval $[t_i, t_{i+1}]$, then we can use a simple Brownian bridge interpolation between the points $\widehat{X}_i$ and $\widehat{X}_{i+1}$. We use (5) to compute the minima $\widehat{Y}_i$ using uniformly distributed random values U_i:

$$\widehat{Y}_{i+1} = \frac{1}{2}\left(\widehat{X}_{i+1} + \widehat{X}_i - \sqrt{(\widehat{X}_{i+1} - \widehat{X}_i)^2 - 2hb_i^2 U_i}\right)$$

Then

$$\widehat{\tau}_m = \begin{cases} \frac{1}{2}(t_{i-1} + t_i), & \text{if } i = \min\left\{k \mid \widehat{Y}_k \leq B\right\} \text{ exists} \\ T & \text{otherwise.} \end{cases}$$

We call this method the *minimum* method.

The third method is a variant of the minimum method that uses probabilities instead of random variables. If we express τ as $\tau = \rho \wedge T$, where $\rho = \inf\{u \mid X_u \leq B\}$ is the unbounded variant of τ, we can write

$$\mathbb{E}[\tau] = \int_0^T t f_\rho(t) dt + T\,\mathbb{P}[\tau = T],$$

where $f_\rho(t)$ is the probability density function of ρ. We can discretely approximate this expectation by

$$\widehat{\mathbb{E}}[\widehat{\tau}] = \sum_{i=1}^{K} \frac{t_{i-1} + t_i}{2} P_i + T P_{K+1},$$

where $P_i, i = 1, \ldots, K$ is the probability of the process crossing B in the time interval $(t_{i-1}, t_i]$ for the first time, and $P_{K+1} = 1 - \sum_{i=1}^{K} P_i$ is the probability of the process not crossing the barrier until T. If

$$p_i = \mathbb{P}[\widehat{Y}_i \leq B] = \exp\left(-\frac{2(\widehat{X}_{i-1} - B)(\widehat{X}_i - B)}{b_{i-1}^2 h}\right)$$

is the probability of crossing B in $[t_{i-1}, t_i]$, given that $\widehat{X}_{i-1}, \widehat{X}_i \geq B$, we can express $P_i, i = 1, \ldots, K+1$ as

$$P_i = \prod_{j=1}^{i-1}(1 - p_j)\, p_i \quad \text{for } i \leq K$$

$$P_{K+1} = \prod_{j=1}^{K}(1 - p_j)$$

Using this method, which we call the *probability* method, we get an estimate of the expected first passage time from every sample path, and we then combine these estimates using Monte Carlo averaging.

The methods described above work in a standard Monte Carlo setting and also with the multilevel Monte Carlo algorithm without any adjustments. However, since

we intend to use these methods primarily with the multilevel Monte Carlo methods, we can work out some further improvements. Recall that on most levels of the multilevel Monte Carlo calculations we are estimating the expected value of the difference $\widehat{P}_l - \widehat{P}_{l-1}$. We compute $\widehat{P}_l$ from a sample path generated with the timestep h_l and using the Brownian increments (W_i), and $\widehat{P}_{l-1}$ from a sample path generated with the timestep $h_{l-1} = Mh_l$, and using the same Brownian increments. The methods of calculating $\widehat{P}_l$ and $\widehat{P}_{l-1}$ on a same level therefore differ, and we distinguish them by writing $\widehat{P}_l^f$ for the method using the finer timesteps h_l, and $\widehat{P}_{l-1}^c$ for the method using the coarse timesteps h_{l-1}. Then, we can modify one or both of these methods to reduce the variance of the estimator $\widehat{Y}_l$.

Recall that $\mathrm{Var}[\widehat{Y}_l] = \frac{1}{N_l}\mathrm{Var}[\widehat{P}_l^f - \widehat{P}_{l-1}^c]$. We can write the variance of $\widehat{P}_l^f - \widehat{P}_{l-1}^c$ as

$$\mathrm{Var}[\widehat{P}_l^f - \widehat{P}_{l-1}^c] = \mathrm{Var}[\widehat{P}_l^f] + \mathrm{Var}[\widehat{P}_{l-1}^c] - 2\,\mathrm{Cov}[\widehat{P}_l^f, \widehat{P}_{l-1}^c].$$

Therefore, increasing the positive correlation between $\widehat{P}_l^f$ and $\widehat{P}_{l-1}^c$ would reduce the variance of $\widehat{Y}_l$.

It is important to note that the goal here is not to improve the accuracy of the estimate $\mathbb{E}[\widehat{P}_{l-1}^c]$, but only to reduce the variance $\mathrm{Var}\left[\widehat{P}_l - \widehat{P}_{l-1}\right]$. In fact, in order to retain the benefits of the multilevel Monte Carlo method, it is essential to ensure that $\mathbb{E}[\widehat{P}_l^c] = \mathbb{E}[\widehat{P}_l^f]$, so that $\mathbb{E}[\widehat{Y}] = \mathbb{E}[\widehat{P}_L]$.

To reduce the variance of our methods, we adapt the ideas of Giles (2008b) and use additional points for the coarse path estimate, without changing its expectation. In the *minimum* method of estimating the expected first passage time we generate the minimum of the time interval $[t_i, t_{i+1}]$ using the values at the end points of the interval $\widehat{X}_i$ and $\widehat{X}_{i+1}$. We now improve the algorithm to reduce the variance of $\widehat{Y}_l$. For estimating $\widehat{\tau}$ on the coarse path $\widehat{P}_{l-1}^c$, we will generate a middle point $\widehat{X}_{i+\frac{1}{2}}$ at time $t_{i+\frac{1}{2}} = t_i + \frac{h}{2}$.

$$\widehat{X}_{i+\frac{1}{2}} = \frac{1}{2}\left(\widehat{X}_i + \widehat{X}_{i+1} - b_i D_i\right), \tag{11}$$

where $D_i \sim N\left(\frac{1}{2}\left(\widehat{X}_i + \widehat{X}_{i+1}\right), h\right)$. We can generate D_i from the increments of the increments of the Brownian motion used on the fine path; if we use the notation $\frac{\Delta}{2}W_i$ to denote the Brownian increment on the fine path at time $t_i = ih$, then $D_i = \frac{\Delta}{2}W_{i+\frac{1}{2}} - \frac{\Delta}{2}W_i = (W_{i+1} - W_{i+\frac{1}{2}}) - (W_{i+\frac{1}{2}} - W_i)$ is independent from the Brownian increment $\Delta W_i = \frac{\Delta}{2}W_{i+\frac{1}{2}} + \frac{\Delta}{2}W_i = (W_{i+1} - W_{i+\frac{1}{2}}) + (W_{i+\frac{1}{2}} - W_i)$, which was used

to generate the coarse sample path. In this case, the minimum $\widehat{Y}^c_{i+1}$ becomes

$$\widehat{Y}^c_{i+1} = \min \left\{ \begin{array}{l} \frac{1}{2}\left(\widehat{X}_{i+\frac{1}{2}} + \widehat{X}_i - \sqrt{(\widehat{X}_{i+\frac{1}{2}} - \widehat{X}_i)^2 - hb_i^2 U_{i1}}\right), \\ \frac{1}{2}\left(\widehat{X}_{i+1} + \widehat{X}_{i+\frac{1}{2}} - \sqrt{(\widehat{X}_{i+1} - \widehat{X}_{i+\frac{1}{2}})^2 - hb_i^2 U_{i2}}\right) \end{array} \right\}.$$

We can reuse the realizations U_{i1} and U_{i2} of uniformly distributed random variables that were already used in estimating the first passage time for the finer path. Note that the diffusion coefficient b_i is the same for both minima; although we could use the coefficient $b_{i+\frac{1}{2}} = b(\widehat{X}_{i+\frac{1}{2}}, t_{i+\frac{1}{2}})$ calculated from the generated middle point of the Brownian bridge, we decide not to, in order to preserve the original value of $\mathbb{E}[\widehat{P}^c_{l-1}]$. Furthermore, we still use the middle point of the time interval $[t_i, t_{i+1}]$ when estimating the actual exit time, regardless of which half of the interval the minimum that first breached the barrier was generated on.

In (11), we could use $+b_i D_i$ instead of $-b_i D_i$ and the distribution of $\widehat{X}_{i+\frac{1}{2}}$ would remain unchanged. However, while the absolute value of the covariance $\mathrm{Cov}[\widehat{P}^f_l, \widehat{P}^c_{l-1}]$ increases whichever method of midpoint construction we choose, the correlation between $\widehat{P}^f_l$ and $\widehat{P}^c_{l-1}$ can be either positive or negative, and this depends on the sign before the term $b_i D_i$. We can use numerical experiments to determine which sign performs better.

We use the same idea to improve the variance of $\widehat{Y}_l$ in the *probability* method. First we generate the midpoint $\widehat{X}_{i-\frac{1}{2}}$ of the interval $[t_{i-1}, t_i]$ using the same formula as before. Then we calculate the probability p_i of the minimum $\widehat{Y}_i$ breaching the barrier B within the time interval $[t_{i-1}, t_i]$ using the generated middle point. Assuming $\widehat{X}_{i-\frac{1}{2}} > B$, let p_{i1} denote the probability that the minimum on the interval $[t_{i-1}, t_{i-\frac{1}{2}}]$ is below B, and let p_{i2} denote the probability that the minimum on the interval $[t_{i-\frac{1}{2}}, t_i]$ falls bellow B. Then

$$p_i = 1 - (1 - p_{i1})(1 - p_{i2}) = p_{i1} + p_{i2} - p_{i1}p_{i2} = p_{i1}(1 - p_{i2}) + p_{i2}.$$

We calculate the probabilities p_{i1} and p_{i2} using the formulae

$$p_{i1} = \exp\left(-\frac{4(\widehat{X}_{i-1} - B)(\widehat{X}_{i-\frac{1}{2}} - B)}{b_{i-1}^2 h}\right),$$
$$p_{i2} = \exp\left(-\frac{4(\widehat{X}_{i-\frac{1}{2}} - B)(\widehat{X}_i - B)}{b_{i-1}^2 h}\right).$$

Again, we must be careful to use the value b_{i-1} for both half-intervals, to avoid changing the expected value of $\widehat{P}^c_l$. Although in the *probability* method the introduction

of another randomly generated point into the calculation of probability might seem wasteful, as it increases the variance of $\widehat{P}^c_{l-1}$, the increase in correlation between $\widehat{P}^f_l$ and $\widehat{P}^c_{l-1}$ offsets the increase in variance, yielding significant improvement in computational complexity required to achieve a given level of accuracy.

The variance-reducing ideas in this subsection can be easily extended to $M = 4$ and other powers of 2, and with a little more work also to other values of M.

5 Numerical results

In this section we present the numerical results achieved with the use of methods outlined in the previous section. We begin by specifying a geometric Brownian motion model that is used to test the results, and state the integral expression of the expected first passage time on a bounded time interval. We then show the results of numerical calculations and compare the efficiency of different methods.

The methods developed in this work are intended to be used to estimate the expected first passing times of a broad class of stochastic processes, such as those that can be defined as solutions to the general SDE (1). However, in our numerical experiments, we limit ourselves to the well-known geometric Brownian motion process. Our main motivation for this is the existence of relatively simple expressions for the expected first passage time for such process, which we will use to verify the stochastic methods of estimating the expected first passage times, developed in this work. We also prefer the geometric Brownian motion over the simpler arithmetic Brownian motion, as with the latter the derivative $\frac{\partial b}{\partial X} = 0$ and thus the Euler-Maruyama and Milstein discretisation schemes coincide.

Let S_t be a solution to the SDE

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

that starts at S_0. From (4) we know that the first passage time $\tau = \inf\{u \mid S_u \leq B\}$ is distributed as

$$\tau \sim IG\left(\frac{\log B - \log S_0}{\mu - \frac{1}{2}\sigma^2}, \frac{(\log B - \log S_0)^2}{\sigma^2}\right).$$

Therefore, the probability density function of τ is

$$f_\tau(t) = \frac{\log S_0 - \log B}{\sigma\sqrt{2\pi t^3}} \exp\left(-\frac{\left(\left(\mu - \frac{1}{2}\sigma^2\right)t - \log B + \log S_0\right)^2}{2t\sigma^2}\right).$$

The expected value of the bounded first passage time $\tau \wedge T$ is

$$\mathbb{E}[\tau \wedge T] = \int_0^T t f_\tau(t) dt + T \left(1 - \mathbb{P}[\tau \leq T]\right), \tag{12}$$

where $\mathbb{P}[\tau \leq T]$ can be derived from (3).

In the numerical computations, we simulated a geometric Brownian motion, starting at $S_0 = 1$, with parameters $\mu = 0.01$ and $\sigma = 0.2$, barrier $B = 0.95$ and time horizon $T = 1$. In this case, the approximate value of $\mathbb{E}[\tau \wedge T]$, calculated numerically using formula (12), is 0.339647. The integral $\int_0^T t f_\tau(t) dt$ evaluates to 0.147347, and the tail probability $\mathbb{P}[\tau \geq T] = 0.1923$. The expected value of the unbounded stopping time is $\mathbb{E}[\tau] = 5.129329$. The accuracy test resulted in

$$\left| \left(\int_0^T f_\tau(t) dt + (1 - \mathbb{P}[\tau \leq T]) \right) - 1 \right| < 10^{-6}.$$

The code for numeric estimation of the expected first passage time is written in Matlab (version R2010b). It implements the algorithms described in section 4, which are inspired by and adapted from the code written by Giles (2008b). The numeric results were produced by a 64-bit computer with 2 GB of RAM and Intel® Core™2 Duo processor running at 2.7 GHz. The functions that test the numerical estimates accept a `seed` parameter, with which they seed the pseudo-random generator used to simulate sample paths. In all results presented in this work, we used `seed = 0`, so the results are perfectly reproducible, barring differences in Matlab implementation and CPU floating point accuracy.

scheme	method	result	±	std. err.	accuracy
Euler	simple	0.345678	±	0.018789	1.775536%
	minimum	0.337484	±	0.020992	0.636954%
	probability	0.346429	±	0.019502	1.996622%
Milstein	simple	0.374553	±	0.019773	10.277032%
	minimum	0.337979	±	0.020851	0.491098%
	probability	0.348202	±	0.019504	2.518619%

Table 1: Multilevel Monte Carlo results with $M = 2$ and accuracy $\varepsilon = 0.01$.

Table 1 lists the results of evaluating the multilevel Monte Carlo algorithm with $M = 2$ and target accuracy of $\varepsilon = 0.01$, and Table 2 lists the number of samples evaluated in each level (up to level 5). We can see that the *simple* method requires more than an order of magnitude more samples to converge to the target accuracy

scheme	method	N_0	N_1	N_2	N_3	N_4	N_5
Euler	simple	81536	63097	41334	26627	15943	9302
	minimum	4548	2472	1101	603	186	126
	probability	1228	656	314	130	100	100
Milstein	simple	40647	30923	19941	12475	7406	4289
	minimum	4549	2680	1110	692	305	100
	probability	1376	733	314	127	100	100

Table 2: Multilevel Monte Carlo results with $M = 2$ and accuracy $\varepsilon = 0.01$: the number of sample paths simulated at each level (truncated).

than the *minimum* method, and that the *probability* method requires several times less sample paths still. We can also see that there is a lot of improvement in terms of the speed of convergence between the *simple* method with Euler-Maruyama discretisation, and with the Milstein scheme. This difference is rather insignificant for the two more efficient methods. As the other example shows, the relative inaccuracy of the *simple* method with Milstein discretisation is not unusual; what is unusual is the unexpected accuracy of the *simple* method using the Euler-Maruyama scheme.

scheme	method	result	±	std. err.	accuracy
Euler	simple	0.347857	±	0.004379	2.417210%
	minimum	0.339660	±	0.004634	0.003651%
	probability	0.341165	±	0.004510	0.446832%
Milstein	simple	0.349916	±	0.007610	3.023236%
	minimum	0.339454	±	0.004226	0.056960%
	probability	0.342410	±	0.004362	0.813472%

Table 3: Multilevel Monte Carlo results with $M = 4$ and accuracy $\varepsilon = 0.002$.

Table 3 shows the results of the multilevel Monte Carlo method with $M = 4$ and target accuracy $\varepsilon = 0.002$, and Table 4 lists the number of samples simulated for each level. We see that in this case, both *simple* methods are significantly less accurate than the other two methods. Furthermore, we can see that the gains from using the Milstein scheme are more pronounced now that the target accuracy is higher.

Table 5 lists the results of evaluating a standard Monte Carlo simulation. The number of samples and the number of timesteps on each path were adjusted so that the final standard error was comparable to the standard error of the results of multilevel

scheme	method	N_0	N_1	N_2	N_3	N_4	N_5
Euler	simple	1697000	1142075	479084	172941	63961	23066
	min.	202503	99410	20928	8381	2010	
	prob.	50950	30934	5515	1307	276	
Milstein	simple	1217803	804260	327398	116602	42758	16362
	min.	147106	76803	15691	5317	158	
	prob.	45851	28258	4712	763	116	

Table 4: Multilevel Monte Carlo results with $M = 4$ and accuracy $\varepsilon = 0.002$: the number of sample paths simulated at each level (truncated).

scheme	method	result	$\pm$	std. err.	accuracy
Euler	simple	0.402321	$\pm$	0.005198	18.452537%
	minimum	0.339156	$\pm$	0.005025	0.144571%
	probability	0.339587	$\pm$	0.004856	0.017725%
Milstein	simple	0.401927	$\pm$	0.005193	18.336623%
	minimum	0.338361	$\pm$	0.005019	0.378696%
	probability	0.338849	$\pm$	0.004851	0.235001%

Table 5: Results of evaluating a standard Monte Carlo simulation of different algorithms of estimating the expected first stopping time. We used 50000 sample paths and 100 timesteps on each path.

Monte Carlo algorithm with $M = 4$. We needed 50000 sample paths and 100 timesteps on each path, yielding the total cost proportional to 5000000. For comparison, the computational complexities $C = \sum_{l=0}^{L} N_l M^l$ of the multilevel algorithms are listed in Table 6. We see that the multilevel Monte Carlo *minimum* and the probability methods offer significant gains in efficiency over the standard Monte Carlo versions of these methods.

scheme	method	C
Euler	simple	97637588
	minimum	1985935
	probability	417230
Milstein	simple	48735867
	minimum	1086110
	probability	312803

Table 6: The computational complexities of multilevel Monte Carlo algorithms, evaluated with $M = 4$ and $\varepsilon = 0.002$.

Figure 1 compares the asymptotic behavior of the *probability* method using Milstein scheme simulated by the standard and the multilevel Monte Carlo algorithms. The top left plot shows $\log_M V_l$ as a function of l. A slope of -1 would mean that $\log_M V_l = -l + O(1)$, which is equivalent to $V_l = O(M^{-l}) = O(h_l)$. The slope in Figure 1 is less than -1, therefore V_l has convergence better than $O(h)$. By Theorem 4.0.1, $\beta > 1$ implies that the computational complexity $C = O(\varepsilon^{-2})$. We can see this on the bottom right plot, where $C * \varepsilon^2$ remains constant as $\varepsilon \to 0$. The top right plot shows the behavior of $\log_M \left|\widehat{Y}_l\right|$. We see that the slope is approximately -1, which means that the bias $\mathbb{E}[\widehat{P}_l - \widehat{P}_{l-1}] = O(h_l)$, implying first order weak convergence. The bottom right plot shows the number of paths for each level N_l decreasing exponentially as level increases. It also shows the growing computational complexity as $\varepsilon \to 0$. The bottom right plot, as already mentioned, shows the behavior of computational cost as ε decreases.

We can compare the behaviour of the *probability* method using the Milstein discretisation with the *minimum* method using the Milstein scheme (Figure 2) and the *probability* method using the Euler-Maruyama discretisation scheme (Figure 3). We can see that the plots imply $\beta = 1$ for both methods, which means that their computational complexity is $O(\varepsilon^{-2}(\log \varepsilon)^2)$. This is better than plain Monte Carlo, but worse than what we can achieve with the *probability* method. In the top right plot

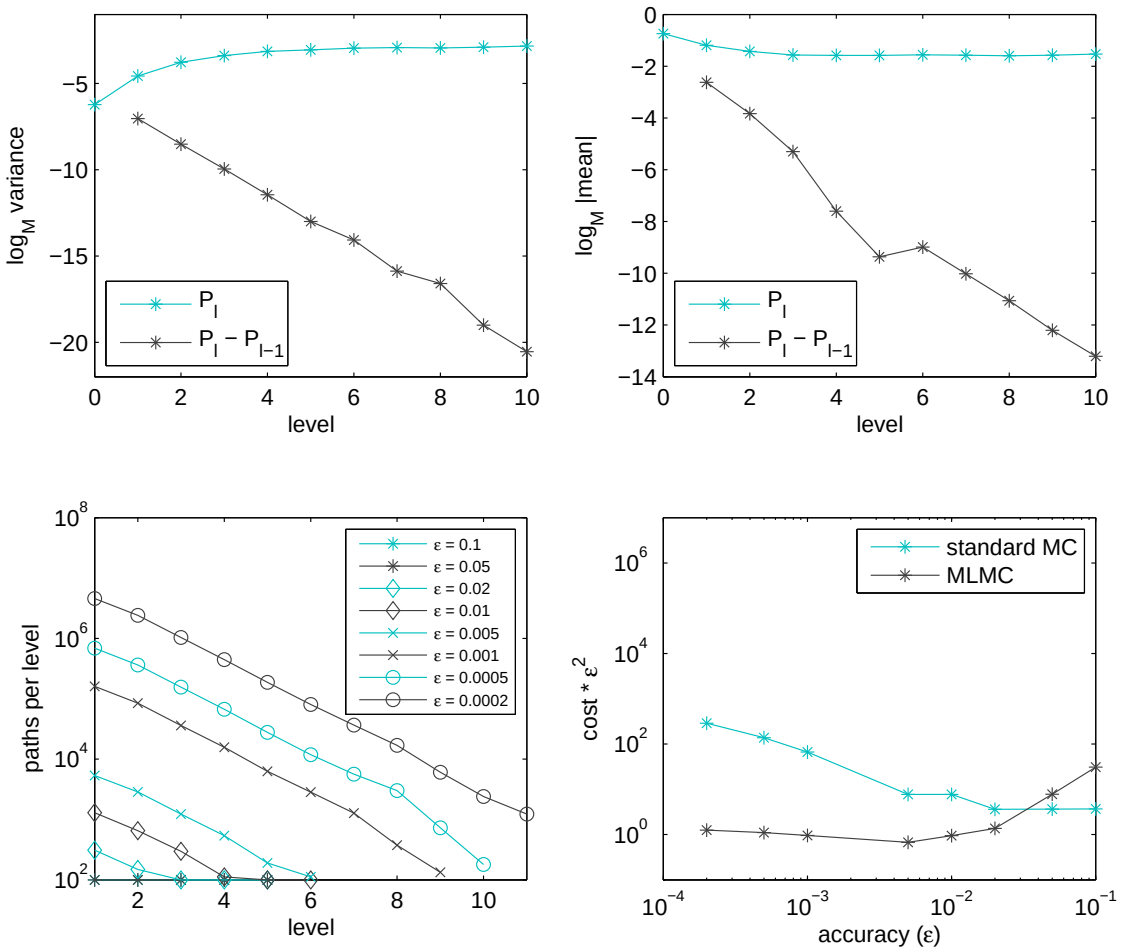

Figure 1: Asymptotic behavior of the *probability* method with Milstein discretisation scheme with standard and multilevel Monte Carlo algorithms for $M = 2$.

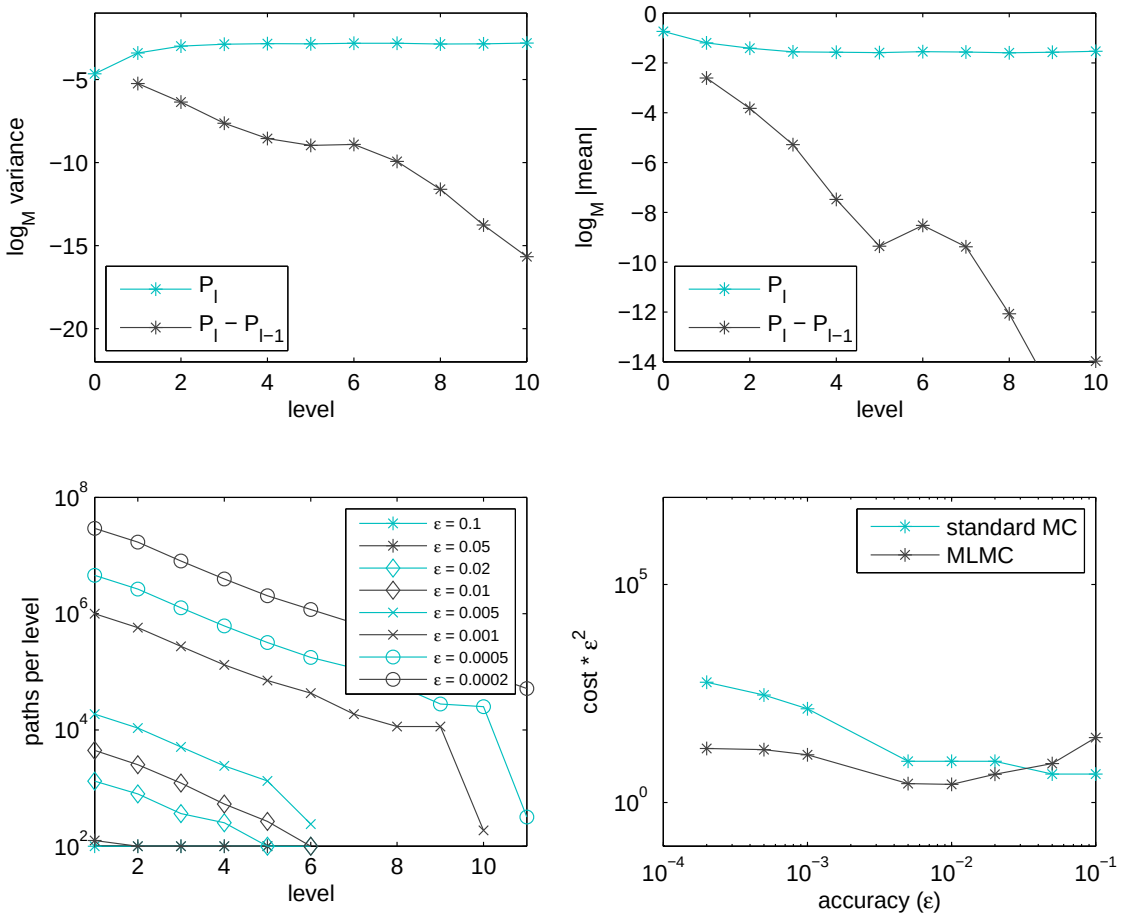

Figure 2: Asymptotic behavior of the *minimum* method with Milstein discretisation scheme with standard and multilevel Monte Carlo algorithms for $M = 2$.

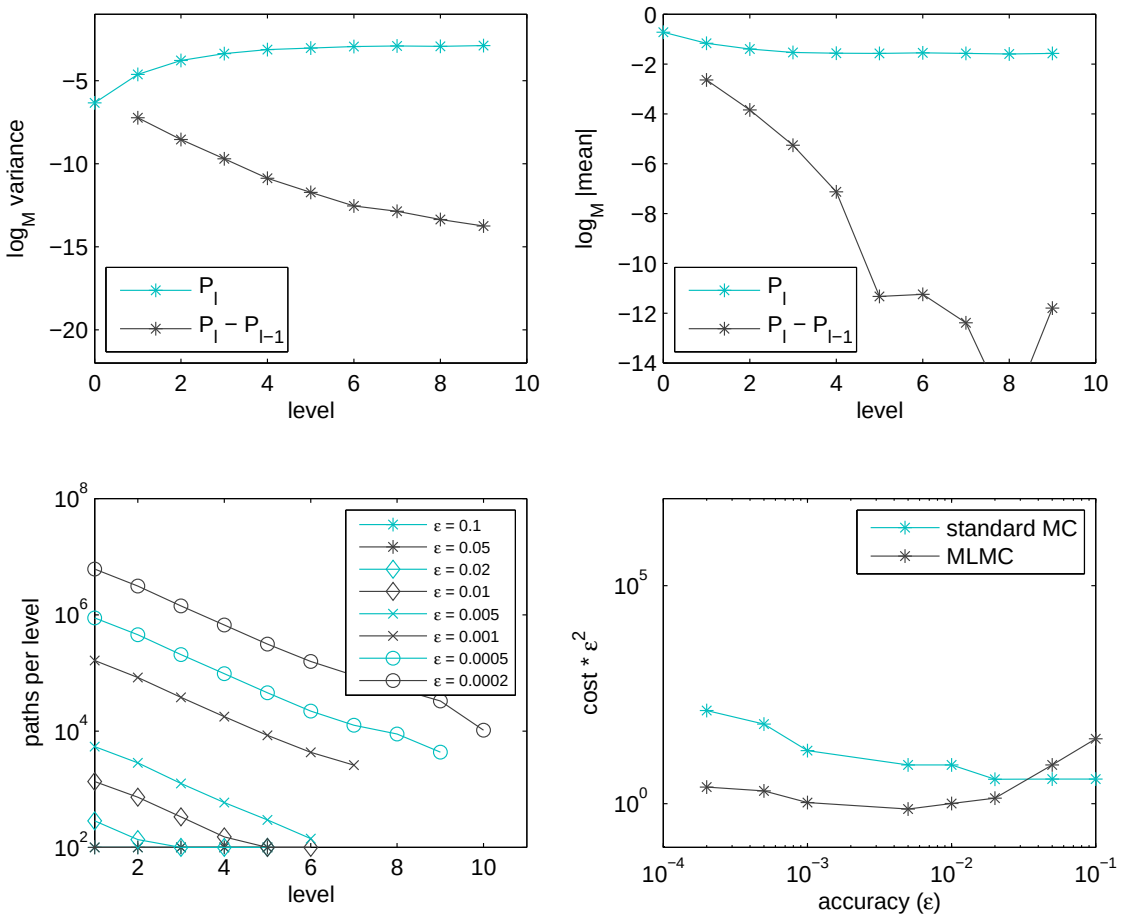

Figure 3: Asymptotic behavior of the *probability* method with Euler-Maruyama discretisation scheme with standard and multilevel Monte Carlo algorithms for $M = 2$.

of Figure 3 we can see $\left|\widehat{Y}_l\right|$ falling very close to 0 and then increasing again. This is why we use the last two levels to estimate the remaining bias and to test wether the multilevel method has already converged.

6 Conclusion

In this work, we have developed an efficient method of estimating expected first passage times of stochastic processes. Due to the specific nature of this problem, we have used the Milstein discretisation technique, which achieves first order strong convergence. We have also adapted the the multilevel Monte Carlo method to our problem, and have implemented additional variance-reducing techniques based on Brownian bridge interpolation, in order to further improve convergence.

We have tested our techniques using geometric Brownian motion simulation, and

have achieved significant performance gains over naïve implementations. In (2011), Higham et al. achieved an improvement in computational cost from $O(\varepsilon^{-4})$ using plain Monte Carlo algorithm, to $O(\varepsilon^{-3})$ using the multilevel Monte Carlo method with Euler-Maruyama discretisation. In our work, numerical results indicate that our most advanced algorithm, the *probability* method, achieves $O(\varepsilon^{-2})$ computational complexity. Our results indicate that these improvements result from using the Milstein discretisation scheme and using the Brownian bridge interpolation for determining the probability of crossing the barrier.

Closed form solutions for expected first passage times are known only for a small class of stochastic processes. In some cases they can be approximated using low-dimensional integrals, such as in our case of geometric Brownian motion. However, for many cases, no analytic approximations are known, and therefore probabilistic methods of estimating the moments are needed. The technique that we have developed in this work is extensible in many ways. Possible extensions include estimating the expected value of a function of the first passage time, such as calculating the expected discount factor. Our method can also be applied to other types of diffusions, such as Ornstein–Uhlenbeck process and the CEV process. Finally, the algorithm can be adapted to include a time-dependent barrier $B = B(t)$.

Appendix - Matlab Code

In this appendix, we list the main implementation of the multilevel Monte Carlo algorithm as outlined in this work. Only the relevant parts of the code are included.

```
% Options:
%       'A' – simulate Arithmetic BM:   dX_t = a dt + b dW_t
%       'G' – simulate Geometric BM:    dX_t = a X_t dt + b X_t dW_t    (default)
%        –
%       'M' – use Milstein discretization (default) (has no effect if 'A' is set)
%       'E' – use Euler–Maruyama discretization (default if 'A' is set)
%        –
%       's' – use simple estimate of the first passage time
%       'm' – use generated minima to estimate the first passage time
%       'p' – estimate the expected first passage time using probabilities (default)
%        –
%       'R' – not included in this listing
%

function [tau, N, std_err] = mlmc(X0, a, b, B, T, M, epsilon, options)

if nargin < 8
```

```
    options = '';
end
if any(options == 'A')      % simulate Arithmetic BM
  f = @(dW, X, h) X + a * h + b * dW;
  g = @(X) b;               % diffusion coefficient is constant
else
  g = @(X) b * X;           % diffusion coefficient is proportional to X
  if any(options == 'E')     % simulate GBM using Euler–Maruyama discretization
    f = @(dW, X, h) X .* (1 + a * h + b * dW);
  else            % simulate Geometric BM using Milstein discretization
    f = @(dW, X, h) X .* (1 + a * h + b * dW + 0.5 * b^2 * (dW.^2 - h));
  end
end

% Multi-level Monte Carlo algorithm by Mike Giles, see
% papers OPRE08 and MCQMC06 and code on his website
L = 0;
N_initial = 100;    % the number of paths to generate initially, estimate variance
converged = false;

A = zeros(3,1);      % this structure keeps the results for indivual levels
%       A = [  N_0,  N_1,  N_2, ...,  N_L ;
%              Y_0,  Y_1,  Y_2, ...,  Y_L ;
%            Y2_0, Y2_1, Y2_2, ..., Y2_L ]
% where N_l is the number of MC paths on level l, Y_l is the sum of the
% difference of estimated stopping times on the fine and on the coarse
% levels, and Y_l is the sum of the squares of said differences.

while ~converged
  dif = mlmc_sim(X0, f, g, B, T, M, L, N_initial, options);

  A(1, L+1) = N_initial;
  A(2, L+1) = dif(1);
  A(3, L+1) = dif(2);
  % biased estimates of variances
  vars = A(3, :) ./ A(1, :) - (A(2, :) ./ A(1, :)).^2;

  % Estimate the optimal number of sample paths for each level. The
  % optimal N_l is proportional to sqrt(Var_l * h_l).
  N = max(N_initial, ceil(2 * sqrt(vars ./ (M.^(0:L))) * sum(sqrt(vars .* ...
      (M.^(0:L)))) / epsilon^2));
  % We have calculated new optimal numbers of paths for each level, maybe
  % we should have more paths on some level than we have now.
  for l = 0:L
    dN = N(l+1) - A(1, l+1);
    if dN > 0       % If we need more paths, let's ...
      dif =  mlmc_sim(X0, f, g, B, T, M, l, dN, options);  % ... simulate them
      A(1, l+1) = A(1, l+1) +  dN;         % ... and update the results.      |
      A(2, l+1) = A(2, l+1) + dif(1);
      A(3, l+1) = A(3, l+1) + dif(2);
    end
  end
  % Test for convergence, to see if we need to use another level, or have
  % we reached the desired accuracy.
```

```
  if L > 1 && M^L ≥ 16       % if less than 3 levels, or if not enough
        % points on the fine sample path, we haven't converged yet.

    if M^L > 1024       % stop if we have more than 2^10 points on the fine
      converged = true;
    else
      converged = max( abs(A(2, L) / A(1, L)) / M, ...
                       abs(A(2, L+1) / A(1, L+1)) ...
                     ) < (M-1) * epsilon / sqrt(2);
    end
  end
  L = L + 1;
end
% sample variances (unbiased estimates)
vars = (A(3, :) - A(2, :).^2 ./ A(1, :)) ./ (A(1, :) - 1);
std_err = sqrt(sum(vars ./ A(1, :)));       % standard error of the estimated tau
tau = sum(A(2, :) ./ A(1, :));
end

% f - the function that calculates the next X along the path
%   parameters:  dW - a row of Brownian increments
%                X - a row of current values of the process
%                h - the current timestep
%   returns:  X_new - a row of new values of the process
% g - the function that calculates the diffusion coefficient
%   parameters:   X - a row of current values of the process
%   returns:      b - a number or a row of diffusion coefficients
function [dif, fine] = mlmc_sim(X0, f, g, B, T, M, l, N, options)
K_f = M^l;          % number of points on the fine path
K_c = K_f / M;      % number of points on the coarse path
h_f = T / K_f;      % timestep of the fine path
h_c = T / K_c;      % timestep of the coarse path
sqrt_h_f = sqrt(h_f);
sum_f = 0;          % sum of simulated stopping times
sum_dif = 0;
square_sum_f = 0;   % sum of squares of simulated stopping times
square_sum_dif = 0;

if any(options == 's')      % use simple estimate for the first passage time
  simple = true;
  minimum = false;
  probability = false;
elseif any(options == 'm')  % generate minima to estimate the first passage time
  simple = false;
  minimum = true;
  probability = false;
else                        % estimate expected first passage time using probabilities
  simple = false;
  minimum = false;
  probability = true;
end

for n_ = 0:10000:N          % simulate at most 10000 paths at once
  n = min(10000, N - n_);
```

```
125   if n == 0
126     break
127   end
128   X_f = X0 * ones(1, n);  % the current value of the process alonge the fine path
129   X_c = X_f;              % the current value of the process along the coarse path
130   tau_f = T * ones(1, n);     % estimated first passage time along the fine path
131   tau_c = T * ones(1, n);     % estimated first passage time along the coarse path
132   if probability
133     tau_f = zeros(1, n);
134     tau_c = zeros(1, n);
135     P_f = ones(1, n);         % probability that the barrier was not yet hit
136     P_c = ones(1, n);
137   end
138   if l == 0
139     dW = sqrt_h_f * randn(1, n);
140     X_old = X_f;
141     X_f = f(dW, X_f, h_f);
142
143     if simple
144       tau_f(X_f < B) = T/2;
145     elseif minimum
146       U = rand(1, n);
147       mins = 0.5 * (X_f + X_old - sqrt((X_f - X_old).^2 - 2 * h_f * g(X_old).^2 .* ...
              log(U)));
148       tau_f(mins < B) = T/2;
149     else    % probability
150       p = exp(-2 * (X_old - B) .* (X_f - B) ./ (g(X_old).^2 * h_f));
151       p(X_f < B) = 1;
152       tau_f = p * T/2 + (1 - p) * T;
153     end
154   else
155     for i = 1:K_c
156       dW_f = sqrt_h_f * randn(M, n);
157       if minimum
158         U = rand(M, n);
159       end
160
161       for m = 1:M
162         X_old_f = X_f;
163         X_f = f(dW_f(m, :), X_f, h_f);
164
165         if simple
166           tau_f(X_f < B) = min(((i - 1) * M + m - 0.5) * h_f, tau_f(X_f < B));
167         elseif minimum
168           mins_f = 0.5 * (X_f + X_old_f - sqrt((X_f - X_old_f).^2 - 2 * h_f * ...
                  g(X_old_f).^2 .* log(U(m, :))));
169           tau_f(mins_f < B) = min(((i - 1) * M + m - 0.5) * h_f, tau_f(mins_f < B));
170         else    % probability
171           p_f = exp(-2 * (X_old_f - B) .* (X_f - B) ./ (g(X_old_f).^2 * h_f));
172           p_f(X_f < B) = 1;
173
174           tau_f = tau_f + P_f .* p_f * ((i - 1) * M + m - 0.5) * h_f;
175           P_f = P_f .* (1 - p_f);
176         end
```

```
177         end
178
179         X_old_c = X_c;
180         dW_c = sum(dW_f);
181         X_c = f(dW_c, X_c, h_c);
182
183         if simple
184           tau_c(X_c < B) = min((i - 0.5) * h_c, tau_c(X_c < B));
185         elseif minimum || probability
186           % We generate the midpoints for the coarse path for two
187           % special cases, M = 2 and M = 4.
188           b = g(X_old_c);
189           if M == 2
190             % generated point at the half of the interval
191             X_half_c = 0.5 * (X_old_c + X_c - b .* (dW_f(2, :) - dW_f(1, :)));
192           elseif M == 4
193             % generated point at the half of the interval
194             X_half_c = 0.5 * (X_old_c + X_c - b .* (dW_f(3, :) + dW_f(4, :) - ...
                    dW_f(1, :) - dW_f(2, :)));
195             % generated point at the quarter of the interval
196             X_quarter_1_c = 0.5 * (X_old_c + X_half_c - b .* (dW_f(2, :) - dW_f(1, :)));
197             % generated point at three quarters of the interval
198             X_quarter_3_c = 0.5 * (X_half_c + X_c - b .* (dW_f(4, :) - dW_f(3, :)));
199           end
200
201           if minimum
202             if M == 2
203               mins_1 = 0.5 * (X_half_c + X_old_c - sqrt((X_half_c - X_old_c).^2 - ...
                      h_c * b.^2 .* log(U(1, :))));
204               mins_2 = 0.5 * (X_c + X_half_c - sqrt((X_c - X_half_c).^2 - h_c * b.^2 ...
                      .* log(U(2, :))));
205               mins_c = min(mins_1, mins_2);
206             elseif M == 4
207               mins_1 = 0.5 * (X_quarter_1_c + X_old_c - sqrt((X_quarter_1_c - ...
                      X_old_c).^2 - 0.5 * h_c * b.^2 .* log(U(1, :))));
208               mins_2 = 0.5 * (X_half_c + X_quarter_1_c - sqrt((X_half_c - ...
                      X_quarter_1_c).^2 - 0.5 * h_c * b.^2 .* log(U(2, :))));
209               mins_3 = 0.5 * (X_quarter_3_c + X_half_c - sqrt((X_quarter_3_c - ...
                      X_half_c).^2 - 0.5 * h_c * b.^2 .* log(U(3, :))));
210               mins_4 = 0.5 * (X_c + X_quarter_3_c - sqrt((X_c - X_quarter_3_c).^2 - ...
                      0.5 * h_c * b.^2 .* log(U(4, :))));
211               mins_c = min(min(mins_1, mins_2), min(mins_3, mins_4));
212             else
213               U_c = rand(1, n);
214               mins_c = 0.5 * (X_c + X_old_c - sqrt((X_c - X_old_c).^2 - 2 * h_c * ...
                      b.^2 .* log(U_c)));
215             end
216               tau_c(mins_c < B) = min((i - 0.5) * h_c, tau_c(mins_c < B));
217           else    % probability
218             if M == 2
219               p1_c = exp(-4 * (X_old_c - B) .* (X_half_c - B) ./ (g(X_old_c).^2 * h_c));
220               p2_c = exp(-4 * (X_half_c - B) .* (X_c - B) ./ (g(X_old_c).^2 * h_c));
221
222               p_c = p1_c .* (1 - p2_c) + p2_c;
```

```
223             p_c(X_half_c < B | X_c < B) = 1;
224           elseif M == 4
225             p1_c = exp(-8 * (X_old_c - B) .* (X_quarter_1_c - B) ./ (g(X_old_c).^2 ...
                    * h_c));
226             p2_c = exp(-8 * (X_quarter_1_c - B) .* (X_half_c - B) ./ ...
                    (g(X_old_c).^2 * h_c));
227             p3_c = exp(-8 * (X_half_c - B) .* (X_quarter_3_c - B) ./ ...
                    (g(X_old_c).^2 * h_c));
228             p4_c = exp(-8 * (X_quarter_3_c - B) .* (X_c - B) ./ (g(X_old_c).^2 * ...
                    h_c));
229
230             p_c = 1 - (1 - p1_c) .* (1 - p2_c) .* (1 - p3_c) .* (1 - p4_c);
231             p_c(X_quarter_1_c < B | X_half_c < B | X_quarter_3_c < B | X_c < B) = 1;
232           else
233             p_c = exp(-2 * (X_old_c - B) .* (X_c - B) ./ (g(X_old_c).^2 * h_c));
234             p_c(X_c < B) = 1;
235           end
236           tau_c = tau_c + P_c .* p_c * (i - 0.5) * h_c;
237           P_c = P_c .* (1 - p_c);
238         end
239       end
240     end
241     if probability
242       tau_f = tau_f + P_f * T;
243       tau_c = tau_c + P_c * T;
244     end
245   end
246
247   sum_dif = sum_dif + sum(tau_f - tau_c);
248   sum_f = sum_f + sum(tau_f);
249
250   square_sum_dif = square_sum_dif + sum((tau_f - tau_c).^2);
251   square_sum_f = square_sum_f + sum(tau_f.^2);
252 end
253 dif = [sum_dif, square_sum_dif];
254 fine = [sum_f, square_sum_f];
255 if l == 0
256   dif = fine;
257 end
258 end
```

References

Giles, Michael B. 2008a. *Multilevel Monte Carlo path simulation*, Operations Research **56**, no. 3, 607–617.

_____. 2008b. *Improved multilevel Monte Carlo convergence using the Milstein scheme*, Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer-Verlag.

Higham, Desmond J., Xuerong Mao, Mikolaj Roj, Qingshuo Song, and George Yin. 2011. *Mean exit times and the multi-level Monte Carlo method*, Technical Report 5, University of Strathclyde.

Glasserman, Paul. 2004. *Monte Carlo methods in financial engineering*, Springer-Verlag.

Kloeden, Peter E. and Eckhard Platen. 1992. *Numerical solution of stochastic differential equations*, Springer-Verlag.

Devroye, Luc. 1986. *Non-Uniform Random Variate Generation*, Springer-Verlag, New York.

Bedingham, D. J. 2008. *State reduction dynamics in a simplified QED model*, Journal of Physics A: Mathematical and Theoretical **41**, 495–205.

Trabelsi, Donia. 2007. *Risk Averse Venture Capital and Timing of IPO under Profit Flows Uncertainty*, Panthéon-Sorbonne University.

Mauer, David C. and Steven H. Ott. 1995. *Investment Under Uncertainty: The Case of Replacement Investment Decisions*, Journal of Financial and Quantitative Analysis **30**, no. 4.

Jia, Ya and Jia-rong Li. 1996. *Transient properties of a bistable kinetic model with correlations between additive and multiplicative noises: Mean first-passage time*, Physical Review E **53**, no. 6, 5764–5768.

Gitterman, M. 2000. *Mean first passage time for anomalous diffusion*, Physical Review E **62**, no. 5, 6065–6070.

Dias, Marco Antonio Guimarães. *First Hitting Time and Expected Discount Factor*. Available online at `http://www.puc-rio.br/marco.ind/hittingt.html` [2011, June 22]

Gauthier, Laurent. 2002. *Excursions Height- and Length-Related Stopping Times, and Application to Finance*, Advances in Applied Probability **34**, no. 4, 846–868.

Linetsky, Vadim. 2004. *Lookback options and diffusion hitting times: A spectral expansion approach*, Finance and Stochastics **8**, no. 3, 373–398.

Lo, C. F., H. M. Tang, K. C. Ku, and C. H. Hui. 2009. *Valuing Time-Dependent CEV Barrier Options*, Journal of Applied Mathematics and Decision Sciences **2009**.

Atiya, Amir F. and Steve A. K. Metwally. 2005. *Efficient Estimation of First Passage Time Density Function for Jump-Diffusion Processes*, SIAM Journal on Scientific Computing **26**, 1760–1775.