Application of simplest random walk algorithms for pricing barrier options

April 2019

1 Introduction

Barrier option contracts are one of the oldest exotic derivatives. The contracts will be activated or deactivated by hitting a certain barrier, depending on the types of the contracts. We can get closed form solutions only under some certain situations. In most cases, we have to use a numerical way to get the price, which can be a challenge. In this paper we assume that the underlying assets are modelled via multidimensional stochastic differential equations (SDEs) and we consider European-type barrier options.

"Ordinary" numerical methods for SDEs are based on a time discretization. They ensure smallness of time increments at each step, but might not ensure smallness of space increments. In some literature, adaptive methods of each time step are proposed so that the next state of a Markov chain approximating in the weak sense the SDEs' solution remains in the bounded domain with probability one. This leads to a decrease of the time step when the chain is close to the boundary so that values could be approximated more accurately. In this report, we will review the method proposed in [N. Milstein and Tretyakov, 2003]. When the chain is far away from the boundary, we update the chain with a constant time step h, following the weak Euler schema. When the chain is close to the boundary, falling to a narrow boundary zone, we make an intermediate step of the random walk. On this auxiliary step we "flip a coin" to decide whether to terminate the chain on the boundary or jump back to the domain and continue the random walk. This construction is based on the idea of linear interpolation for the the solution of SDEs.

[Krivko and Tretyakov, 2012] demonstrates how to use this method to pricing barrier options, via specific examples. This report focus on two of them, presenting the detailed algorithms and the corresponding simulation results. In the following, we will firstly present the general objective problem and the proposed algorithms in section 2. In section 3, we will give firstly the Libor rate model on which barrier cap and barrier swaption are based. Then we will give more details on the two barrier options and show their corresponding simulation results.

2 Algorithm

In order to deal with these three different examples, we introduce the basic tool,simplest random walks for stopped diffusions. The basic idea is the price of barrier options with underlying modelled by a diffusion process can be expressed as

$$u(t,x) = E\left[\varphi\left(\tau, X_{t,x}(\tau)\right) Y_{t,x,1}(\tau) + Z_{t,x,1,0}(\tau)\right], \tag{1}$$

where $X_{t,x}(s), Y_{t,x,y}(s), Z_{t,x,y,z}(s), s \geq t$, is the solution of the Cauchy problem for the system of SDEs.

$$dX = (b(s,X) - \sigma(s,X)\mu(s,X))ds + \sigma(s,X)dw(s), \quad X(t) = x \tag{2}$$
$$dY = c(s,X)Yds + \mu^{\top}(s,X)Ydw(s), Y(t) = y \tag{3}$$
$$dZ = g(s,X)Yds + F^{\top}(s,X)Ydw(s), \quad Z(t) = z \tag{4}$$

Then we can apply the weak explicit Euler approximation with the simplest simulation

$$\begin{aligned} X_{t,x}(t+h) &\approx X = x + h(b(t,x) - \sigma(t,x)\mu(t,x)) + h^{1/2}\sigma(t,x)\xi, \\ Y_{t,x,y}(t+h) &\approx Y = y + hc(t,x)y + h^{1/2}\mu^{T}(t,x)y\xi, \\ Z_{t,x,y,z}(t+h) &\approx Z = z + hg(t,x)y + h^{1/2}F^{T}(t,x)y\xi, \end{aligned} \tag{5}$$

Now we introduce the properties and models near the boundary, let $S_{t,h} \in G$ on the layer t: we say that x$\in$S$_{t,h}$ if the distance from $x \in G$ to the boundary ∂G is equal to or grater than $\lambda\sqrt{h}$ then x is outside the boundary zone which indicates that all the realizations of the random variable X belong to G.

In order to add some restricts on the approximation so we can make sure that X does exit the domain G, we need to build a special construction, which can be derived as an algorithm. Let x belongs to $S_{t,h}$, we denote $x^\pi \in \partial G$ the projection of the point x on the boundary of the domain G. We also denote $n(x^\pi)$ the unit vector of internal normal to ∂G at x^π. We introduce the random vector $X^\pi_{x,h}$ taking two values x^π and $x + h^{\frac{1}{2}}(x^\pi)$ with certain probabilities. The former value means the vector stops on the boundary, the latter one means it jumps back in the domain and keep doing the simulation. We construct a Markov chain which stops when it reaches the boundary at random step $\varkappa$. The details of the algorithm are in the following algorithm part. The second algorithm is a simplification of the first one. We will terminate the random walk once it goes into the boundary domain.

Algorithm 1: Algorithm of weak order one

```
STEP 0 : $X'_0 = x_0, Y_0 = 1, Z_0 = 0, k = 0$;
STEP 1 : If $X'_k \notin S_{t_k,h}$, then $X_k = X'_k$ and then go to STEP 3.
         If $X'_k \in S_{t_k,h}$, then either $X_k = X'^{\pi}_k$ with probability $p_{X'_k,h}$
         or $X_k = X'_k + h^{1/2}\lambda n(X'^{\pi}_k)$ with probability $q_{X'_k,h}$;
STEP 2 : Simulate $\xi_k$ and find $X'_{k+1}, Y_{k+1}, Z_{k+1}$ according to (5) for
         $t = t_k, x = X_k, y = Y_k, z = Z_k, \xi = \xi_k$.;
STEP 4 : If $k + 1 = M$, Stop and $\varkappa = M, X_\varkappa = X'_M, Y_\varkappa = Y_M, Z_\varkappa = Z_M$,
         otherwise $k = k + 1$ and return to STEP 1.
```

Algorithm 2: Algorithm of weak order 1/2

```
STEP 0 : $X_0 = x_0, Y_0 = 1, Z_0 = 0, k = 0$;
STEP 1 : If $X_k \notin S_{t_k,h}$, then go to STEP 2.
         If $X_k \in S_{t_k,h}$, then Stop and $\varkappa = k, \bar{X}_\varkappa = X^\pi_k, Y_\varkappa = Y_k, Z_\varkappa = Z_k$ ;
STEP 2 : Simulate $\xi_k$ and find $X_{k+1}, Y_{k+1}, Z_{k+1}$ according to (5) for
         $t = t_k, x = X_k, y = Y_k, z = Z_k, \xi = \xi_k$.;
STEP 3 : If $k + 1 = M$, Stop and $\varkappa = M, \bar{X}_\varkappa = X_M, Y_\varkappa = Y_M, Z_\varkappa = Z_M$,
         otherwise $k = k + 1$ and return to STEP 1.;
```

We also give the traditional Monte Carlo method here, as described in Algorithm 3.

Algorithm 3: Simple Monte Carlo method

```
STEP 0 : $X_0 = x_0, Y_0 = 1, Z_0 = 0, k = 0$;
STEP 1 : If $X_k \notin G$, then Stop and $\varkappa = M, \bar{X}_\varkappa = X_M, Y_\varkappa = Y_M, Z_\varkappa = Z_M$.
         Else go to STEP 2 ;
STEP 2 : Simulate $\xi_k$ and find $X_{k+1}, Y_{k+1}, Z_{k+1}$ according to (5) for
         $t = t_k, x = X_k, y = Y_k, z = Z_k, \xi = \xi_k$. and return to STEP 1;
```

3 Experiments

3.1 LIBOR Market Model

We will use LIBOR rate as underlying asset for the following barrier options. We have the following simple replication argument for LIBOR rate,

$$L(t, T, T + \delta) = \frac{1}{\delta}\left(\frac{P(t, T)}{P(t, T + \delta)} - 1\right) \tag{6}$$

For simplicity, here we fix as equidistant finite set of tenor dates,

$$T_0 < \cdots < T_N = T^*, \quad T_i = i\delta, \quad i = 0, \ldots, N \tag{7}$$

where $\delta = (T^* - T_0)/N$ denotes the fixed length of the interval between tenor dates. We note $L^i(t) := L(t, T_i, T_{i+1})$, and in the case of LIBOR Market Model the arbitrage-free dynamics of $L^i(t)$ under the forward measure $Q^{T_{k+1}}$ associated with the numeraire $P(t, T_{k+1})$ can be written as the following system of

SDEs:

$$\frac{dL^i(t)}{L^i(t)} = \begin{cases} \sigma_i(t) \sum_{j=k+1}^{i} \frac{\delta L^j(t)}{1+\delta L^j(t)} \rho_{i,j}\sigma_j(t)dt + \sigma_i(t)dW_i^{T_{k+1}}(t), i > k, t \leq T_k \\ \sigma_i(t)dW_i^{T_{k+1}}(t), i = k, t \leq T_i \\ -\sigma_i(t) \sum_{j=i+1}^{k} \frac{\delta L^j(t)}{1+\delta L^j(t)} \rho_{i,j}\sigma_j(t)dt + \sigma_i(t)dW_i^{T_{k+1}}(t), i < k, t \leq T_i \end{cases} \tag{8}$$

where $W^{T_{k+1}} = \left(W_0^{T_{k+1}}, \dots, W_{N-1}^{T_{k+1}}\right)^\top$ is an N-dimentional correlated Wiener process, with correlation structure defined as

$$E\left[W_i^{T_{k+1}}(t)W_j^{T_{k+1}}(t)\right] = \rho_{i,j}, \quad i,j = 0,\dots,N-1 \tag{9}$$

and $\sigma_i(t), i = 0,\dots,N-1$, are instantaneous volatilities which assume here to be deterministic bounded functions. Given the correlation matrix ρ with elements $\rho_{i,j}$, we can use Cholesky decomposition to construct easily the N-dimentional process W. In what follows we assume this correlation function:

$$\rho_{i,j} = \exp\left(-\beta \left|T_i - T_j\right|\right) \tag{10}$$

3.2 Barrier cap/floor

We consider Monte Carlo evaluation of barrier options written on a single underlying. We use a knock-out caplet for illustration.

An Interest Rate Cap is a financial tool to protect the holder from the low floating rate and can be protected from high ones. When we consider the underlying rate as the LIBOR rate, the valuation of the battier cap equals to the price of European barrier option on equity with zero interest rate multiplying the term $\delta P(t, T_{i+1})$. So we can have a closed-form solution. Then we can make the following algorithm to realize the simulation.

3.2.1 Algorithm

We simulate the log dynamics of the Libor rate in order to preserve its positivity. The dynamics of $L^i(s)$ is

$$\frac{dL^i(s)}{L^i(s)} = \sigma(s)dW_i^{T_{i+1}}(s), s \leq T_i. \tag{11}$$

We add the dynamics with this equation

$$dZ = F(s, L^i)dW_i^{T_i+1}(s), Z(0) = 0, \tag{12}$$

with

$$F(s, L^i) = -\sigma_i(s)\frac{\partial}{\partial L^i}\tilde{V}_{caplet}(s, L^i(s)). \tag{13}$$

We choose a time step $h > 0$ so that $M = \frac{T_i}{h}$ is an integer and determine the initial value of lnL_0^i and Z_0($L^i(0)$ and 0, respectively). Then we use the weak Euler form so we get

$$\log L_{k+1}^i = \log L_k^i - \frac{1}{2}(\sigma_i(t_k))^2 h + \sigma_i(t_k)\sqrt{h}\xi_{k+1} \tag{14}$$

$$Z_{k+1} = Z_k + F(s, L_k^i)\sqrt{h}\xi_{k+1} \tag{15}$$

where ξ_k are independent random variables distributed by the law $P(\xi = \pm 1) = \frac{1}{2}$.

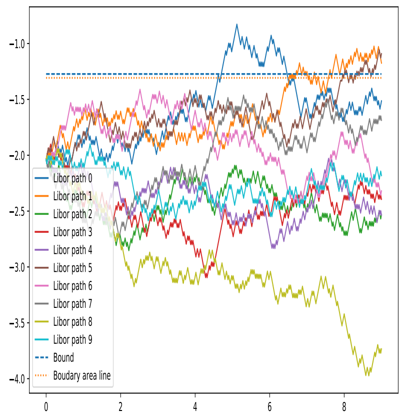
Then we formalize the "out of the boundary" condition as

$$\log L_k^i < \log H + \frac{1}{2}\sigma_i^2(t_k)h - \sigma_i(t_k)\sqrt{h}, \tag{16}$$

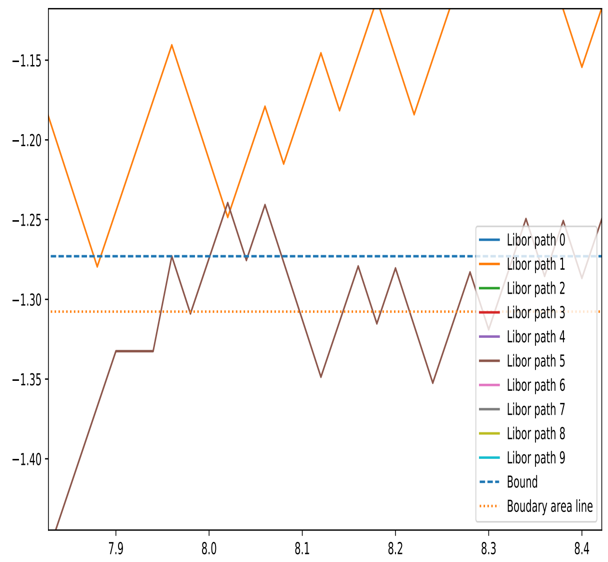
where λ_k satisfies

$$\lambda_k\sqrt{h} = -\frac{1}{2}\sigma_i^2(t_k)h + \sigma_i(t_k)\sqrt{h} \tag{17}$$

We build a random walk generated by (13). We check the condition (16) at every time to see if we will continue the random walk. If it holds, we do the same process to generate the next step. If it doesn't, we do an extra step to determine whether we stop or we go back to the domain at the position $\log L_k^i - \lambda_k\sqrt{h}$. Also, we will stop if the time arrives the time boundary M.



(a) Several paths of L^9



(b) Behaviours near the bound

Figure 1: Sample paths of L^9 simulated with Algorithm 1

Our goal is to evaluate the expectation

$$\begin{aligned}\tilde{V}_{\text{caplet}}(0) &= E^{\mathrm{Q}^{T_{i+1}}}\left[\left(L^i\left(T_i\right)-K\right)_+\chi\left(\theta>T_i\right)+Z(\tau)\right] \\ &\approx E^{\mathrm{Q}^{T_{i+1}}}\left[\left(\exp\left(\ln L^i_{\varkappa}\right)-K\right)_+\chi(\varkappa=M)+Z_{\varkappa}\right]\end{aligned} \tag{18}$$

3.2.2 Numerical results

Here we present some results of numerical tests of Algorithm 1 and Algorithm 2 for pricing the barrier caplet. We use the following parameters in the experiments : $i=9, K=1\%, H=28\%, \delta=1, L^9(0)=13\%$. The volatility $\sigma_i(t)$ is assumed to be constant at 25%.

Firstly we give an visual illustration of Algorithm 1. Figure 1a gives several samples of L^9 simulated by using Algorithm 1 with $h=0.02$, and Figure 1b zooms in its behaviours near the bound. For path 5, at $t=7.92$, it should fall in the boundary zone with normal update, while according to the rule of Algorithm 1, it retreats from the boundary zone. Similarly for $t=7.94$. While at $t=7.96$, the command from Algorithm 1 is to project itself to the bound. Thus we see that it falls exactly on the bound at $t=7.96$.

Figure 2 gives the monte carlo simulation results with $h=0.02$. We repeat 100000 experiments for Algorithm 1, Algorithm 2 and simple Monte Carlo method (Algorithm 3). We can see that with more simulations, the bias of all the three methods tend to be smaller. The error obtained with Algorithm 2 is much smaller than the other two method. And we also notice that the result of Algorithm 2 is always below the true value, since it increases the probability of knocking-out.

We also ran the experiments with different h, whose results are shown in Figure 3. Generally, with a bigger h, a bigger error is produced. And for the most choices of h, the error of Algorithm 2 is the smallest.

3.3 Barrier swaption

In this section we consider Monte Carlo evaluation of under the LMM. Without loss of generality, we concentrate on a knock-out receiver swaption with the first reset date T0. A knock-out swaption has the structure as a stan- dard swaption except that if the underlying swap rate is above a barrier level Rup at any time before T0 then the swaption expires worthless. The price of the knock-out swaption at time $t=0$ under the

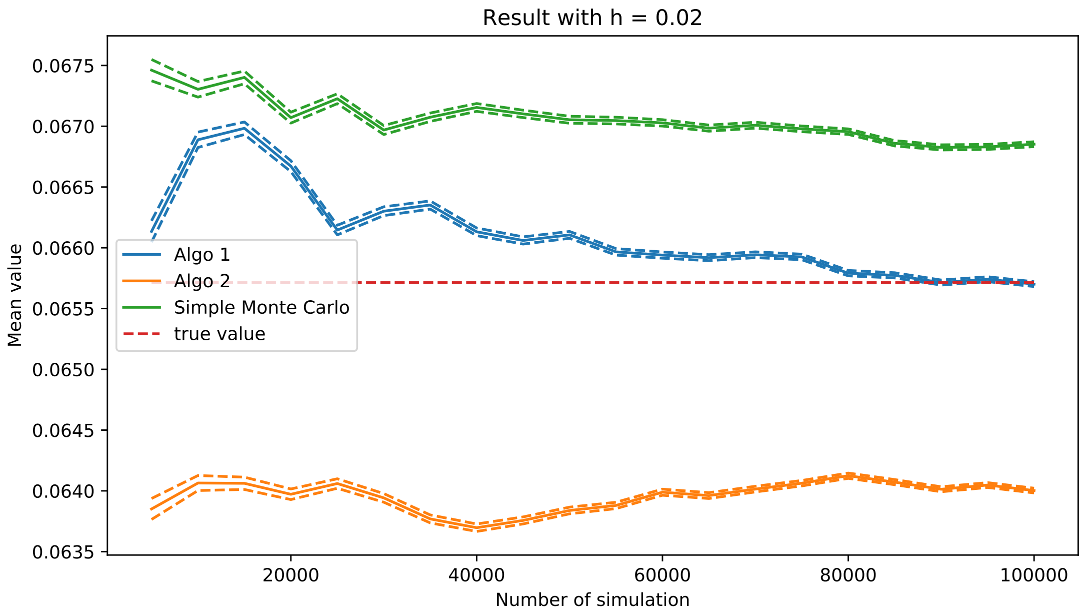


Figure 2: Barrier caplet pricing with $h = 0.02$

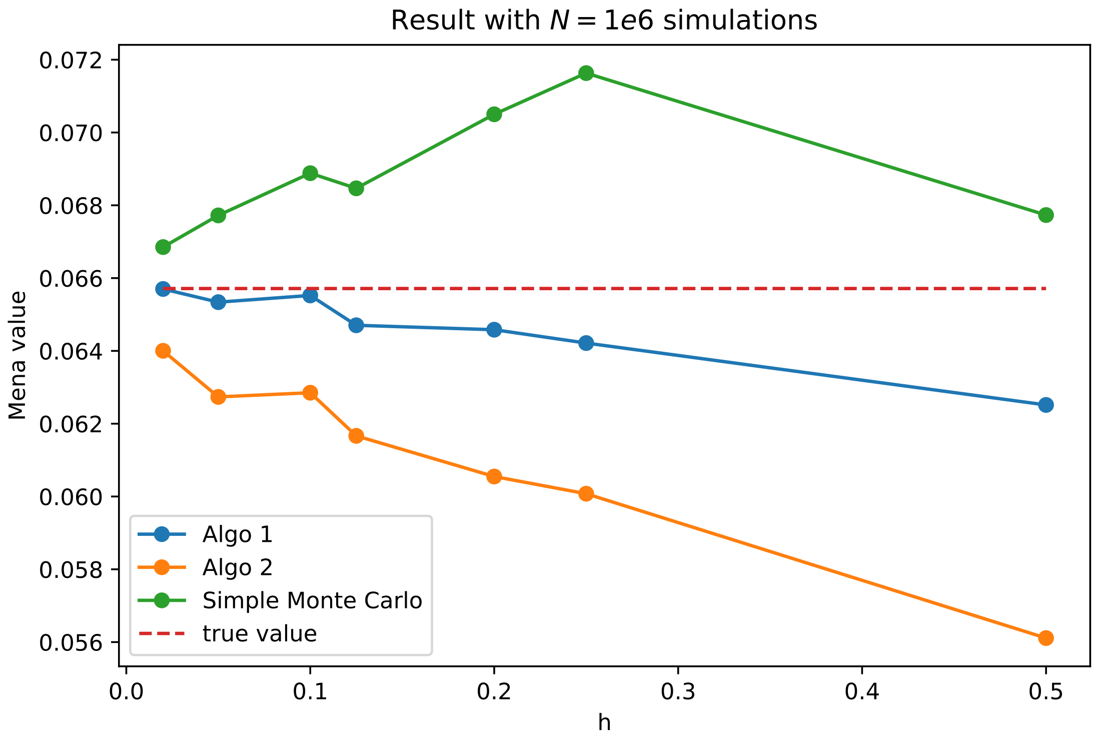


Figure 3: Barrier caplet pricing with different values of h

forward measure Q^{T_0} is given by :

$$V_{\text{swaption}}(0) = P(0, T_0)\, E^{Q^{T_0}} \left[\delta \left(R_{\text{swap}}(T_0) - K\right)_+ \sum_{j=1}^{N} P(T_0, T_j)\, \chi(\theta > T_0) \right] \tag{19}$$

where θ is the first exit time of the process $R_{swap}(s), s \geq 0$ from the interval $(0, R_{up})$. We note in LMM model under Q^{T_0} are given by:

$$\frac{dL^i(t)}{L^i(t)} = \sigma_i(t) \sum_{j=0}^{i} \frac{\delta L^j(t)}{1+\delta L^j(t)} \rho_{i,j} \sigma_j(t) dt + \sigma_i(t) dW_i^{T_0}(t), i = 0, \ldots, N-1 \tag{20}$$

For test purpose, we also use the closed form solution under Swap Market Model as an approximate value.

3.3.1 Algorithm

Here we illustrate the realization of Algorithm 1. Firstly we update $\ln L_K$ according to the following rule with constant h until we reach the boundary zone $S_{t_k,h}$.

$$\begin{aligned} \ln L_{k+1}^i =& \ln L_k^i + \sigma_i(t_k) \sum_{j=0}^{i} \frac{\delta L_k^j}{1+\delta L_k^j} \rho_{ij} \sigma_j(t_k) h \\ & - \frac{1}{2} (\sigma_i(t_k))^2 h + \sigma_i(t_k) \sqrt{h} \sum_{j=0}^{N-1} U_{i,j} \xi_{j,k+1} \\ i =& 0, \ldots, N-1 \end{aligned} \tag{21}$$

where $\xi_{j,k}$ are mutually independent random variables distributed by the law $P(\xi = \pm 1) = 1/2$.

To verify if $\ln L_K$ is in $S_{t_k,h}$, we apply a two-step check method. Firstly, we introduce $\ln L_{k,\mathrm{Max}} = \max_i \ln L_k^i$, and $\ln \hat{L}_{k+1} = \ln L_{k,Max} + \sigma_{Max}^2 hN - \frac{1}{2}\sigma_{Max}^2 h + \sigma_{Max}\sqrt{hN}$, with $\sigma_{Max} = \max_{i,k} \sigma_i(t_k)$. Using the fact $R_{\mathrm{swap}}\left(\hat{L}_{k+1}, \ldots, \hat{L}_{k+1}\right) = \hat{L}_{k+1}$, we can see the current position $\ln L_K$ is inside the domain G if

$$\ln \hat{L}_{k+1} < \ln R_{up} \tag{22}$$

This condition is computationally easy to check. If it is satisfied, we can continue the random walk following 21. If not, we check the following condition

$$\begin{aligned} & R_{swap}\left(L_k^0\left(1 + \sigma_0(t_k)\sigma_{\mathrm{Max},k} h + \sigma_0(t_k)\sqrt{Nh}\right), L_k^1\left(1 + 2\sigma_1(t_k)\sigma_{\mathrm{Max},k} h\right.\right. \\ & \left.\left. + \sigma_1(t_k)\sqrt{(N-1)h}\right), \ldots, L_k^{N-1}\left(1 + N\sigma_{N-1}(t_k)\sigma_{\mathrm{Max},k} h + \sigma_{N-1}(t_k)\sqrt{h}\right)\right) < R_{up} \end{aligned} \tag{23}$$

where $\sigma_{Max,k} = \max_j \sigma_j(t_k)$. If it is true, we continue the normal update unless $k+1 = M$. If both condition failed, it means that we have reached the boundary zone $S_{t_k,h}$. To implement Algorithm 1, 2, we need to find the projection $\ln L_k^\pi$, which is equivalent to find the minimum value of the function

$$|\ln L_k^\pi - \ln L_k|^2 = \left(\ln L_k^{\pi,0} - \ln L_k^0\right)^2 + \cdots + \left(L_k^{\pi,N-1} - \ln L_k^{N-1}\right)^2 \tag{24}$$

subject to the constraint

$$\ln\left(\frac{\prod_{j=0}^{N-1}\left(1+\delta L_k^{\pi,j}\right) - 1}{\delta\left(1 + \sum_{i=0}^{N-2}\prod_{j=i+1}^{N-1}\left(1+\delta L_k^{\pi,j}\right)\right)}\right) = \ln R_{up} \tag{25}$$

Once we obtain the projection, either we stop the chain at $\ln L_k^\pi$ with probability p:

$$p = \frac{\lambda\sqrt{h}}{|\ln L_k^\pi - \ln L_k| + \lambda\sqrt{h}} \tag{26}$$

where $\lambda\sqrt{h} = \sqrt{N}\left(\sigma_{\mathrm{Max}}^2 hN - \frac{1}{2}\sigma_{\mathrm{Max}}^2 h + \sigma_{\mathrm{Max}}\sqrt{hN}\right)$. Or we back to the domain G to the point $\ln L_k + \lambda\sqrt{h}\frac{\overline{\ln L_k^\pi \ln L_k}}{\ln L_k^\pi - \ln L_k|}$ with probability $1-p$, and continue the above steps. The outcome of each trajectory is the point $(t_x, \ln L_x)$.

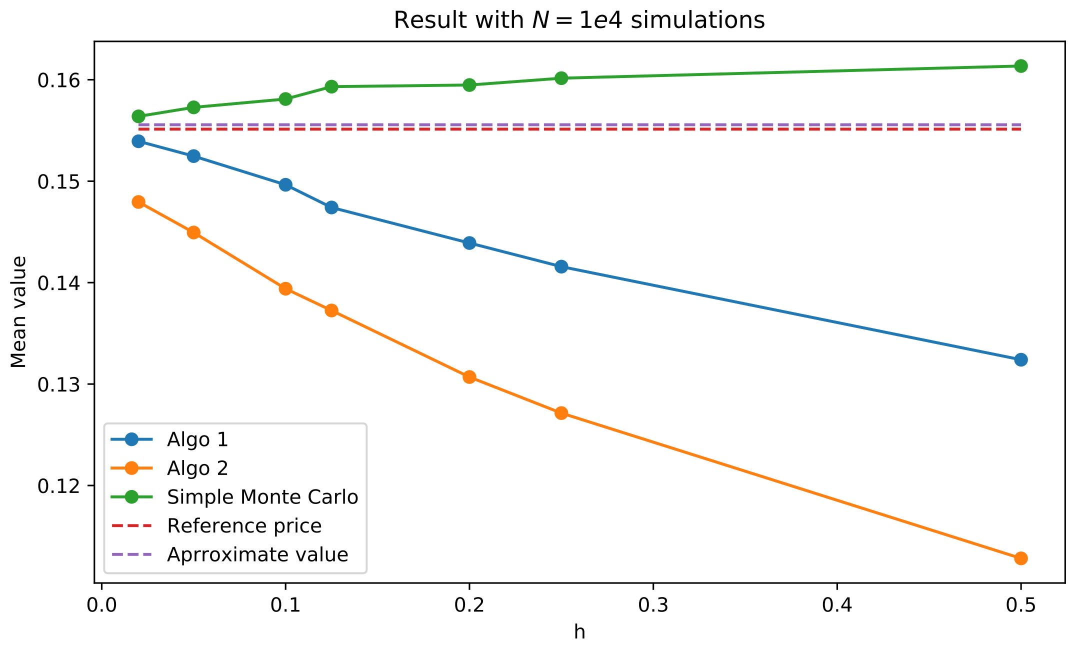


Figure 4: Barrier swaption pricing with different values of h

h	error	mean exit time
0.02	$1.19 * 10^{-3} \pm 1.38 * 10^{-3}$	9.54
0.05	$2.66 * 10^{-3} \pm 1.40 * 10^{-3}$	9.50
0.1	$5.48 * 10^{-3} \pm 1.43 * 10^{-3}$	9.44
0.125	$7.72 * 10^{-3} \pm 1.45 * 10^{-3}$	9.39
0.2	$1.12 * 10^{-2} \pm 1.45 * 10^{-3}$	9.35
0.25	$1.35 * 10^{-2} \pm 1.48 * 10^{-3}$	9.26
0.5	$2.27 * 10^{-2} \pm 1.54 * 10^{-3}$	9.01

Table 1: Performance of Algorithm 1 for the barrier swaption

3.3.2 Numerical results

Here we give some numerical results for pricing a barrier swaption by using the previous algorithms. The initial LIBOR curve is assumed to be flat at 5%. The volatility $\sigma_i(t)$ is constant at 10%, and following parameters are considered: $T_0 = 10, T^\star = 20, K = 0.01, R_{up} = 0.075, \delta = 1, \beta = 0.1$. We firstly run 10^5 simulations with $h = 0.01$. We use the result 0.15513 as the reference price, with the Monte Carlo error $4.38 * 10^{-4}$, which gives half of the size of the confidence interval for the corresponding estimator with probability 0.95. The analytical approximation yields a price of 0.15556. We then run the experiments 10^4 times for each choice of h. Figure 4 gives the result. We can see generally with bigger h, the error is bigger. Unexpectedly, we find that in this specific example, the simple Monte Carlo method performs better than algorithm 1 and algorithm 2 with these h, while better performance could be expected with smaller h for Algorithm 1. Table 1 and table 2 give more details. It is clear that the results demonstrate the expected first order of convergence for algorithm 1 and one second order of convergence for algorithm 2.

h	error	mean exit time
0.02	$7.18 * 10^{-3} \pm 1.45 * 10^{-3}$	9.43
0.05	$1.02 * 10^{-2} \pm 1.46 * 10^{-3}$	9.37
0.1	$1.57 * 10^{-2} \pm 1.50 * 10^{-3}$	9.25
0.125	$1.79 * 10^{-2} \pm 1.52 * 10^{-3}$	9.18
0.2	$2.44 * 10^{-2} \pm 1.54 * 10^{-3}$	9.04
0.25	$2.79 * 10^{-2} \pm 1.56 * 10^{-3}$	8.95
0.5	$4.23 * 10^{-2} \pm 1.61 * 10^{-3}$	8.50

Table 2: Performance of Algorithm 2 for the barrier swaption

References

[Krivko and Tretyakov, 2012] Krivko, M. and Tretyakov, M. (2012). Application of simplest random walk algorithms for pricing barrier options.

[N. Milstein and Tretyakov, 2003] N. Milstein, G. and Tretyakov, M. (2003). The simplest random walks for the dirichlet problem. *Theory of Probability and Its Applications - THEOR PROBAB APPL-ENGL TR*, 47.