Probabilités numériques et méthodes de Monte Carlo

Application of simplest random walk algorithms for pricing barrier options

Caroline Nguyen & Tina Nguyen Truong
Master Probabilités et Finance (ex DEA El Karoui)

Sommaire

1 Introduction

The objective of the article under study is the implementation of the random walk algorithm, similar to the Brownian bridge method, for pricing barrier options within a diffusion model. The pricing of barrier Caps and Swaptions will be performed and compared to a standard Monte Carlo method.

2 Theoretical Aspects

2.1 Random walk algorithm for diffusions with stopping time

2.1.1 Problem Definition

We work on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We consider time t such that $0 \leq t \leq T$ and a domain $G \subset \mathbb{R}^d$ such that $Q = [t_0, T[\times G$ is a cylinder in $\mathbb{R}^{d+1}$. We denote $\Gamma = \bar{Q} \setminus Q$ as the boundary of the domain Q. The process $(w_t, \mathcal{F}_\sqcup)$ is a standard r-dimensional Wiener process. By the Feynman-Kac theorem, we know that the price $u(t,x)$ is a solution to the PDE and is written as:

$$u(t,x) = \mathbb{E}[\varphi(\tau, X_{t,x}(\tau))Y_{t,x,1}(\tau) + Z_{t,x,1,0}(\tau)] \tag{2.1}$$

where $X_{t,x}(s), Y_{t,x,y}(s), Z_{t,x,y,z}(s),\ t \leq s$ are solutions to the Cauchy problem for the following system:

In the SDEs, $b(s,x)$ is a d-dimensional vector, $\sigma(s,x)$ is a d×r matrix, $\mu(s,x)$ and F(s,x) are r-dimensional vectors, and Y(s), Z(s), c(s,X) and g(s,X) are scalars.

We assume that these coefficients and the function $\varphi(t,x)$ defined on Γ and ∂G are sufficiently regular.

We note that $u(t,x)$ does not depend on $\mu(s,x)$ and $F(s,x)$. This allows us to reduce the variance when computing the expectation using the Monte Carlo method.
For example, if we have:

$$\sum_{i=1}^{d} \sigma^{ij} \partial_{x_i} u + u\mu^j + F^j = 0,\ j = 1, ..., r \tag{2.5}$$

then $Var[\varphi(\tau, X_{t,x}(\tau))Y_{t,x,1}(\tau) + Z_{t,x,1,0}(\tau)] = 0$ and $\varphi(\tau, X_{t,x}(\tau))Y_{t,x,1}(\tau) + Z_{t,x,1,0}(\tau) \equiv u(t,x)$. Since the choice of optimal μ and F depends on $u(t,x)$, this is not practical. We will therefore use proxies for $u(t,x)$ to determine the optimal values.

2.1.2 Solutions

To find the solution on the domain Q, we approximate the SDE system using a weak Euler scheme.

$h > 0$ is the discretization step size, $\xi = (\xi^1, ..., \xi^r)^T$, where ξ^i, i=1,...,r are i.i.d. random variables taking the values ± 1 with probability $p = \frac{1}{2}$. This implies that X takes 2^r different values.

To find the solution near the boundary of Q, we first construct the set $S_{t,h}$ as a zone close to the boundary of G, so $S_{t,h} \subset \bar{G}$. $S_{t,h}$ is the set of $x \in G$ such that at least one of the 2^r values of X does not belong to $\bar{G}$. Since $\bar{G}$ is compact, there exists $\lambda > 0$ such that if the distance from x to ∂G is greater than or equal to $\lambda\sqrt{h}$, then $x \notin S_{t,h}$. In other words, if $x \in \bar{G} \backslash S_{t,h}$, then $X \in \bar{G}$.

To construct the solution near the boundary, there are two algorithms with different convergence orders:

Algorithm of weak order one. $\mathcal{O}(h)$ Convergence:

Let $x \in S_{t,h}$, let x^π be the projection of x onto ∂G, let $n(x^\pi)$ be the unit inward normal vector at x^π, $p = p_{x,h}$, and $q = q_{x,h} = 1 - p_{x,h}$ such that:

$$p_{x,h} = \frac{h^{\frac{1}{2}}\lambda}{|x + h^{\frac{1}{2}}\lambda n(x^\pi) - x^\pi|}, \tag{2.9}$$

$$X^\pi_{x,h} = \begin{cases} x^\pi & \text{with probability } p \\ x + h^{\frac{1}{2}}\lambda n(x^\pi) & \text{with probability } q \end{cases}$$

Where $x + h^{1/2}\lambda n(x^\pi) \notin S_{t,h}$ and $p > \frac{1}{2}$. This can be explained by the approximation of a function v in $C^2(\bar{G})$:

$$v(x) = \mathbb{E}v(X^\pi_{x,h}) + O(h) = pv(x^\pi) + qv(x + h^{\frac{1}{2}}\lambda n(x^\pi)) + O(h) \tag{2.10}$$

Algorithm of weak order 1/2. $\mathcal{O}(\sqrt{h})$ Convergence:
This is a simplified version of the previous algorithm. If $x \in S_{t,h}$ then $X^\pi_{x,h}$ only takes as value x^π (ie $X^\pi_{x,h} = x^\pi$ with p=1).

2.2 LIBOR Market Model

This model focuses on the direct modeling of forward rates, the underlying variable for caps and floors. The LIBOR serves as the benchmark forward rate.

For notation, we will use equidistant maturities: $T_0 < ... < T_N = T^*$, $T_i = i\delta$ where i=0,...,N with $\delta = \frac{T^* - T_0}{N}$.
The LIBOR rate $L^i(t) = L(t, T_i, T_{i+1}) = \frac{1}{\delta}(\frac{P(t,T_i)}{P(t,T_{i+1})} - 1)$ is the forward rate at time t for borrowing 1 unit at T_i and repaying $1 + \delta L^i(t)$ at T_{i+1}.
LIBOR dynamics under forward probability are defined as follows :

$$\frac{dL^i(t)}{L^i(t)} = \begin{cases} \sigma_i(t)\sum_{j=k+1}^{i} \frac{\delta L^j(t)}{1+\delta L^j(t)}\rho_{i,j}\sigma_j(t)dt + \sigma_i(t)dW_i^{T_{k+1}}(t) & i > k, t \leq T_k \\ \sigma_i(t)dW_i^{T_{k+1}}(t) & i = k, t \leq T_k \\ -\sigma_i(t)\sum_{j=i+1}^{k} \frac{\delta L^j(t)}{1+\delta L^j(t)}\rho_{i,j}\sigma_j(t)dt + \sigma_i(t)dW_i^{T_{k+1}}(t) & i < k, t \leq T_i \end{cases} \tag{3.3}$$

where $W^{T_{k+1}} = (W_0^{T_{k+1}}, ..., W_{N-1}^{T_{k+1}})^T$ is a standard Brownian motion of dimension N under the probability $Q^{T_{k+1}}$ with the numeraire $P(t, T_{k+1})$. The instantaneous correlation function we will use in the following is given by :

$$\mathbb{E}[W_i^{T_{k+1}}(t)W_j^{T_k+1}(t)] = \rho_{i,j} = e^{-\beta|T_i - T_j|} \tag{3.6}$$

with i,j=0,...,N-1
The $\sigma_i(t)$ are the instantaneous volatilities. They are assumed to be deterministic.

2.3 Application to a cap

As an application, we are going to trade a cap with a barrier on a single underlying using the random walk algorithm. A cap is an interest rate option used to hedge against rising interest rates. It is generally valued as the sum of the positive parts of the caplets. A caplet is an option on rates over a unit interval $[T_i, T_{i+1}[$. So taking a cap can be as simple as taking a caplet.
A knock-out caplet is a caplet that pays the payoff of a simple caplet until a certain barrier H is crossed. When the barrier is breached, the product no longer exists and its value is therefore zero. The value of a knock-out caplet is given by the following expression :

$$V_{caplet}(t) = \delta P(t, T_{i+1})\mathbb{E}^{Q^{T_{i+1}}}[(L^i(T_i) - K)_+ \mathbf{1}_{(\theta > T_i)}|\mathcal{F}_t] \tag{4.1}$$

The term $\delta P(t, T_{i+1})$ comes from the numeraire change where $P(t, T_{i+1})$ is the zero coupon price at t of maturity T_{i+1}. To add the barrier, we add an indicator that the value of the caplet is zero when the barrier is breached. In fact, θ is the first stopping time of $L^i(s)$.
If τ is the first time out of the trajectory $(s, L^i(s))$ of $Q = [t, T_i) \times (0, H)$ then $\tau = \theta_i^T$. Since the caplet covers a unit interval, the instantaneous correlation is zero, so the dynamics of LIBOR can be written as :

$$\frac{dL_i(s)}{L_i(s)} = \sigma_i(s)\, dW_i^{T_{i+1}}(s), \quad s \leq T_i \tag{4.2}$$

With this dynamic, we can find a closed formula for the caplet. This closed formula allows us to study the algorithm accuracy but it could also be useful for variance reduction.

$$\begin{aligned}
&V_{\text{caplet}}(t) = V_{\text{caplet}}(t, L^i(t)) \\
&= \delta P(t, T_{i+1})\{L^i(t)[\Phi(\delta_+(L^i(t)/K, v_i)) - \Phi(\delta_+(L^i(t)/H, v_i))] \\
&\quad - K[\Phi(\delta_-(L^i(t)/K, v_i)) - \Phi(\delta_-(L^i(t)/H, v_i))] \\
&\quad - H[\Phi(\delta_+(H^2/(KL^i(t)), v_i)) - \Phi(\delta_+(H/L^i(t), v_i))] \\
&\quad + KL^i(t)[\Phi(\delta_-(H^2/(KL^i(t)), v_i)) - \Phi(\delta_-(H/L^i(t), v_i))]/H\},
\end{aligned} \tag{4.3}$$

Φ is the distribution function of the normal distribution, $\delta_+(x, v) = \frac{ln(x) + \frac{v^2}{2}}{v}$ et $\delta_-(x, v) = \frac{ln(x) - \frac{v^2}{2}}{v}$ with $v_i^2 = \int_t^{T_i} (\sigma_i(s))^2 ds$. In the following, instead of computing $V_{caplet}(t)$, we will compute :

$$\tilde{V}_{caplet}(t) = \frac{V_{caplet}(t)}{\delta P(t, T_{i+1})} \tag{4.5}$$

2.4 Application to a swaption

The methodology follows the same stochastic framework as for the caplet, while accounting for the multivariate correlation between LIBOR rates. At the exercise date T_0, the swap rate and the zero-coupon bond prices are derived from the simulated LIBORS :

$$R_{\text{swap}}(s) = \frac{1 - 1/\Pi_{j=0}^{N-1}(1 + \delta L^j(s))}{\delta \sum_{i=0}^{N-1} 1/\Pi_{j=0}^{i}(1 + \delta L^j(s))} \tag{6.2}$$

Under the forward measure associated with T_0, the value of a knock-out receiver swaption is given by the expected positive payoff between the simulated swap rate and the strike, weighted by the probability of not having breached the upper barrier :

$$V_{\text{swaption}}(0) = P(0, T_0) E^{Q^{T_0}} \left[\delta \left(R_{\text{swap}}(T_0) - K \right)_+ \sum_{j=1}^{N} P(T_0, T_j) \chi(\theta > T_0) \right] \tag{6.1}$$

The dynamics of each LIBOR rate $L_i(t)$ under Q^{T_0} follow :

$$\frac{\mathrm{d}L^i(t)}{L^i(t)} = \sigma_i(t)\left(\sum_{j=0}^{i}\frac{\delta L^j(t)}{1+\delta L^j(t)}\rho_{i,j}\sigma_j(t)\right)\mathrm{d}t + \sigma_i(t)\mathrm{d}W_i^{T_0}(t), \quad i = 0,\dots,N-1 \tag{6.3}$$

where the instantaneous correlation between Brownian motions is given by $\mathbb{E}(dW_i dW_j) = \rho_{ij}dt$ where $\rho_{ij} = e^{-\beta|T_i-T_j|}$.
Like the knock-out caplet, the swaption has a closed proxy formula which will give us an idea of our simulations accuracy.

$$\begin{aligned} V_{\text{swaption}}(0) = \delta\sum_{j=1}^{N} P(0,T_j)\{&R_{\text{swap}}(0)[\Phi(\delta_+(R_{\text{swap}}(0)/K, v_{R_{\text{swap}}})) - \Phi(\delta_+(R_{\text{swap}}(0)/R_{\text{up}}, v_{R_{\text{swap}}}))] \\ &- K[\Phi(\delta_-(R_{\text{swap}}(0)/K, v_{R_{\text{swap}}})) - \Phi(\delta_-(R_{\text{swap}}(0)/R_{\text{up}}, v_{R_{\text{swap}}}))] \\ &- R_{\text{up}}[\Phi(\delta_+(R_{\text{up}}^2/(KR_{\text{swap}}(0)), v_{R_{\text{swap}}})) - \Phi(\delta_+(R_{\text{up}}/R_{\text{swap}}(0), v_{R_{\text{swap}}}))] \\ &+ KR_{\text{swap}}(0)[\Phi(\delta_-(R_{\text{up}}^2/(KR_{\text{swap}}(0)), v_{R_{\text{swap}}})) - \Phi(\delta_-(R_{\text{up}}/R_{\text{swap}}(0), v_{R_{\text{swap}}}))]/R_{\text{up}}\}, \end{aligned} \tag{6.4}$$

where $\delta_\pm$ remains the same as before and;

$$v_{R_{swap}}^2 = \int_0^{T_i} (\sigma_{R_{swap}}(s))^2 ds$$

with $\sigma_{R_{swap}}(s)$ the instantaneous volatility of the lognormal dynamics of the swap rate. As an approximation, we can use the following formula :

$$v_{R_{\text{swap}}}^{LMM} = \sum_{i,j=0}^{N-1} \frac{\omega_i(0)\omega_j(0)L^i(0)L^j(0)\rho_{ij}}{(R_{\text{swap}}(0))^2}\int_0^{T_0}\sigma_i(s)\sigma_j(s)\mathrm{d}s$$

where

$$\omega_i(0) = \frac{1/\Pi_{j=0}^{i-1}(1+\delta L^j(0))}{\sum_{k=0}^{N-1} 1/\Pi_{j=0}^{k}(1+\delta L^j(0))}$$

3 Model implementation

3.1 Application to a barrier caplet

To ensure the positivity of the simulated rates, we simulate the logarithm of the LIBOR rate, denoted $\ln(L_k^i)$, instead of L_k^i directly. By exponentiating the results, we guarantee strictly positive values.

3.1.1 Solutions

Simultion within the domain $Q = [t, T_i[\times]0, H[$

The weak Euler scheme for the stochastic differential system is given by;

$$\begin{aligned} \ln L_{k+1}^i &= \ln L_k^i - \frac{1}{2}(\sigma_i(t_k))^2 h + \sigma_i(t_k)\sqrt{h}\xi_{k+1} \qquad (4.8) \\ Y_{k+1} &= 1 \\ Z_{k+1} &= Z_k + F(s, L_k^i)\sqrt{h}\xi_{k+1} \qquad (4.9) \end{aligned}$$

where ξ_k are i.i.d Rademacher random variables (taking values ± 1 with equal probability p=$\frac{1}{2}$) and $M = \frac{T_i}{h}$ is an integer.

Boundary handling in Q

We define the boundary zone $S_{t,h}$ as:

$$S_{t,h} = \{L_k^i : lnL_{k+1}^i \geq ln(H) - \lambda_k\sqrt{h}\} \qquad (4.10)$$

with $\lambda_k\sqrt{h} = \sigma_i(t_k)\sqrt{h}$, ensuring a sufficiently wide buffer zone.

Algorithm 2.1 (convergence order $\mathcal{O}(h)$):
The rebound probability is given by:

$$p = \frac{\lambda_k\sqrt{h}}{|ln(H) - ln(L_k^i) + \lambda_k\sqrt{h}|}$$

If $L_k^i \in S_{t,h}$, then:

$$\ln L_{k+1}^i = \begin{cases} \ln H & \text{with probability } p \\ \ln L_k^i - \lambda_k\sqrt{h} - \frac{1}{2}(\sigma_i(t_k))^2 h + \sigma_i(t_k)\sqrt{h}\xi_{k+1} & \text{with probability } q \end{cases}$$

Algorithm 2 (convergence order $\mathcal{O}(\sqrt{h})$): If $L_k^i \in S_{t,h}$ then $\ln(L_{k+1}^i) = \ln(\mathrm{H})$ and the trajectory $(\theta = k+1, ln(L_\theta^i), Z_\theta)$ is the final one. Else, we continue until M and $(\theta = M, ln(L_\theta^i), Z_\theta)$ will be the final trajectory.

The caplet value is estimated as :

$$\tilde{V}_{\text{caplet}}(t) = \mathbb{E}^{Q^{T_{i+1}}}\left[\left(\exp(\ln L_\theta^i) - K\right)_+ \mathbf{1}_{\theta=M} + Z_\theta\right]$$

Variance reduction
To reduce variance, the function F is chosen based on (2.5) :

$$\begin{aligned} F(s, L^i) &= -\sigma_i(s)\frac{\partial}{\partial L^i}\tilde{V}_{\text{caplet}}(s, L^i) \\ &= [\Phi(\delta_+(L^i(t)/K, v_i)) - \Phi(\delta_+(L^i(t)/H, v_i))] \\ &\quad + K[\Phi(\delta_-(H^2/(KL^i(t)), v_i)) - \Phi(\delta_-(H/L^i(t), v_i))]/H \end{aligned}$$

The derivative is computed analytically from the closed-form formula of the barrier caplet. We also implement an antithetic variate method to further reduce variance:

$$\bar{X} = \frac{X + X'}{2}$$

such that X and $X' = -X$ have the same variance. The estimation variance is given by :

$$Var(\bar{X}) = \frac{Var(X) + Cov(X, X')}{2}$$

This is effective whenever $Cov(X, X') < 0$.

3.2 Application to a Barrier Swaption

The target valuation consists of a sum of a zero-coupon bond prices :

$$\sum_{j=1}^{N} P(T_0, T_j) = \sum_{i=0}^{N-1} 1/\Pi_{j=0}^{i}(1 + \delta L^j(t))$$

Assuming a flat yield curve, we use :

$$P(t, T_0) = \frac{1}{(1 + L(t))^{(T_0 - 1)}}$$

Solution on $Q = [t, T_0[\times G$
Where G is the LIBOR rates space corresponding ... The Euler scheme for each LIBOR rate L_i is :

$$\begin{aligned} \ln L_{k+1}^i = \ln L_k^i + \sigma_i(t_k) \sum_{j=0}^{i} \frac{\delta L_k^j}{1 + \delta L_k^j} \rho_{ij} \sigma_j(t_k) h \\ - \frac{1}{2}(\sigma_i(t_k))^2 h + \sigma_i(t_k)\sqrt{h} \sum_{j=0}^{N-1} U_{i,j} \xi_{j,k+1}, \\ i = 0, \ldots, N-1, \end{aligned} \tag{6.6}$$

where $\rho = UU^T$
Boundary Q:
We define a bounding value :

$$\ln \hat{L}_{k+1} = \max_i \ln L_k^i + \sigma_{Max}^2 hN - \frac{1}{2}\sigma_{Max}^2 h + \sigma_{Max}\sqrt{hN} \tag{6.7}$$

$$\begin{aligned} \tilde{R}_{\text{swap}} = R_{\text{swap}} \Bigg(L_k^0 \left(1 + \sigma_0(t_k)\sigma_{Max,k} h + \sigma_0(t_k)\sqrt{Nh}\right), L_k^1 \left(1 + 2\sigma_1(t_k)\sigma_{Max,k} h + \sigma_1(t_k)\sqrt{(N-1)h}\right), \\ \ldots, L_k^{N-1} \left(1 + N\sigma_{N-1}(t_k)\sigma_{Max,k} h + \sigma_{N-1}(t_k)\sqrt{h}\right) \Bigg), \end{aligned} \tag{6.9}$$

with $\sigma_{Max} = \max_{i,k} \sigma_i(t_k)$ then :

$$S_{t,h} = \{L_k : \ln \hat{L}_{k+1} \geq \ln R_{\text{up}} \text{ et } \tilde{R}_{\text{swap}} \geq R_{\text{up}}\}$$

Algorithm 2.1

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

$$\lambda\sqrt{h} = \sqrt{N}\left(\sigma_{\text{Max}}^2 hN - \frac{1}{2}\sigma_{\text{Max}}^2 h + \sigma_{\text{Max}}\sqrt{hN}\right),$$

Construction of $\ln L_k^{\pi} := \left(\ln L_k^{\pi,0}, \ln L_k^{\pi,1}, \ldots, \ln L_k^{\pi,N-1}\right)^T$ the projection of $\ln L_k$ on la the boundary by solving the following minimisation problem :

$$\begin{cases} \min \|\ln L_k^{\pi} - \ln L_k\|^2 = (\ln L_k^{\pi,0} - \ln L_k^0)^2 + \cdots + (\ln L_k^{\pi,N-1} - \ln L_k^{N-1})^2 \\ \text{such that } \ln L_k^{\pi,0} = \ln\left(\frac{R_{\text{up}}\cdot\left(1+\sum_{i=0}^{N-2}\prod_{j=i+1}^{N-1}(1+\delta\ln L_k^{\pi,j})\right)+1}{\prod_{j=1}^{N-1}(1+\delta\ln L_k^{\pi,j})} - \frac{1}{\delta}\right) \end{cases}$$

So if $L_k \in S_{t,h}$ and $u \sim \mathcal{U}([0,1])$ then :

$$\ln L_{k+1}^i = \begin{cases} \ln L_k^{\pi} & \text{if } u < p \\ \ln L_k + \lambda\sqrt{h}\frac{ln(L_k^{\pi})ln(L_k)}{|ln(L_k^{\pi})-ln(L_k)|} & \text{else} \end{cases}$$

If $ln(L_{k+1}) = ln(L_k^{\pi})$ then the trajectory $(\theta = k+1, ln(L_\theta))$ is the final one. Else, we continue until M and $(\theta = M, ln(L_\theta))$ will be the final one.

Variance reduction
We use the antithetic variable method as described previously.

4 Numerical applications

4.1 Python vs C

Python is more readible and easier to code. It includes a large panel of librairies for various tasks. However, it may have a lower performance compared to C++ and is less suitable for applications requiring intensive processing or optimal speed.

C++ has higher performance, is ideal for application requiring fast execution and allows precise control over memory management. But its syntax is more complex and has less libraries compared to Python.

After considering these factors, we decided to implement our project in Python due to its ease of development and the wide availability of libraries for our specific application.

4.2 Caplet

For the caplet, we will compare the results of algorithm 2.1 and algorithm 2.2. Indeed, the results are similar. However, algorithm 2.1 is computationally expensive. The number of Monte Carlo simulations for the results presented below is 10^5. The other parameters are set to: $\sigma = 0.25$, H = 0.28, i = 9, K = 0.01, $L_0^i = 0.13$, and $Z_0 = 0$.

Using formula (4.3) and (4.5) we find that the exact caplet price is around 6,57%. Then, we perform 10^6 simulations and report the results below.

Algorithm 2.1 weak order 1 with kick back reflection

Table 1: Performance of Algorithm 2.1

h	**Error**	**Mean Exit Time**
0.25	$3.99 \times 10^{-4} \pm 5.77 \times 10^{-5}$	8.13
0.2	$2.89 \times 10^{-4} \pm 5.75 \times 10^{-5}$	8.13
0.125	$5.53 \times 10^{-4} \pm 5.63 \times 10^{-5}$	8.14
0.1	$1.43 \times 10^{-4} \pm 5.67 \times 10^{-5}$	8.14
0.0625	$5.62 \times 10^{-5} \pm 5.67 \times 10^{-5}$	8.14
0.03125	$1.22 \times 10^{-4} \pm 5.66 \times 10^{-5}$	8.14

Algorithm 2.2 weak order $\frac{1}{2}$, absorbing barrier

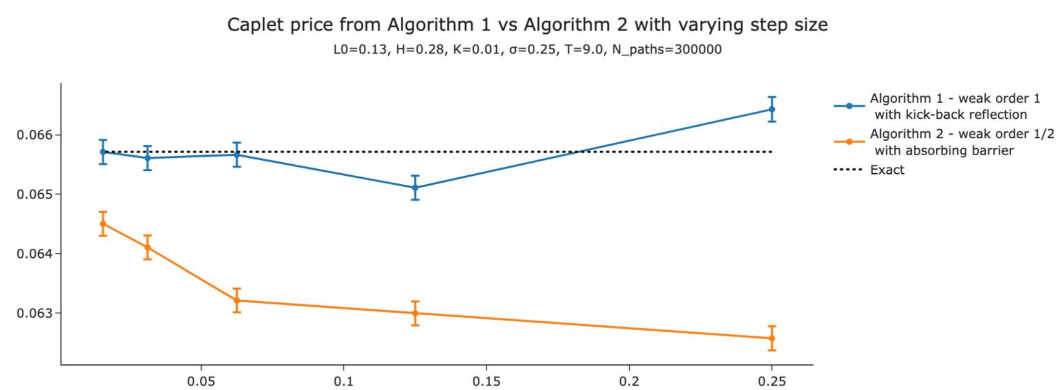
Table 2: Performance of Algorithm 2.2

h	**Error**	**Mean Exit Time**
0.25	$3.27 \times 10^{-3} \pm 5.68 \times 10^{-5}$	7.95
0.2	$3.41 \times 10^{-3} \pm 5.65 \times 10^{-5}$	7.97
0.125	$4.13 \times 10^{-3} \pm 5.46 \times 10^{-5}$	8.03
0.1	$3.11 \times 10^{-3} \pm 5.56 \times 10^{-5}$	8.03
0.0625	$2.53 \times 10^{-3} \pm 5.57 \times 10^{-5}$	8.05
0.03125	$1.89 \times 10^{-3} \pm 5.58 \times 10^{-5}$	8.08

For one million simulations, we observe that the convergence to the analytical result is approximately ten times faster. This improvement is due to the higher order of accuracy of our algorithm.

We observe that the mean exit time in Algorithm 1 is generally higher than in Algorithm 1. This is expected, as Algorithm 2 employs a kick-back reflection mechanism, making it more conservative near the barrier: paths either re-enter the domain or knock out, reducing the likelihood of prolonged survival near the boundary.

As the step size h decreases, the errors become smaller, which is expected since finer discretization provides a more accurate approximation of the continuous-time process. Additionally, the mean exit time increases with smaller h, as trajectories are less likely to make large jumps that would quickly reach the barrier.

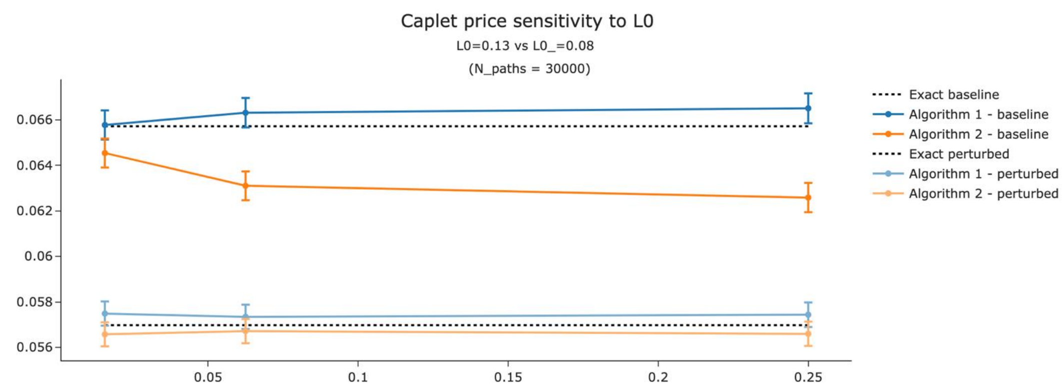


From the plot above, it is obvious that Algorithm 2.1 (kick-back reflection) vastly outperforms Algorithm 2.2 (absorbing barrier). Even at the coarsest time step (h=0.25), Algorithm 2.1's estimates lie almost exactly on the true caplet price (dotted line) and its confidence intervals remain uniformly dmall, reflecting its higher weak order and stability. By contrast, Algorithm 2.2 biased downward-starting near 0.0645- and only inches toward the exact value as h decreases.

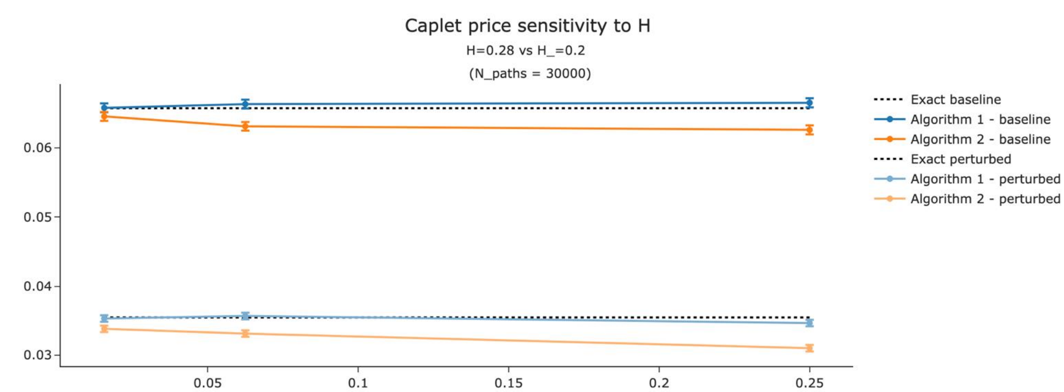
The familiar oscillatory pattern of the binary-tree scheme is visible in Algorithm 2.1's curve but fades as the mesh is refined, whereas Algorithm 2.2's convergence is monotonic yet slow.

In short, Algorithm 2.1 delivers superior accuracy and greater efficiency; Algorithm 2.2 requires prohibitively small steps to even approach the correct price.

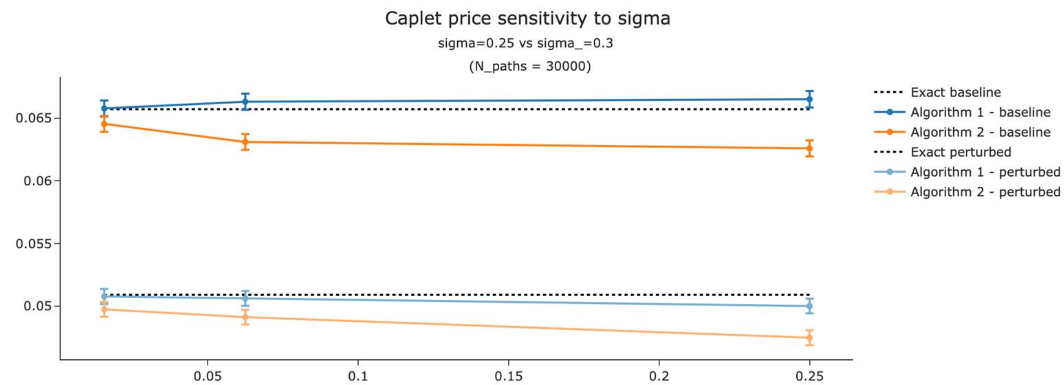
With different parameters



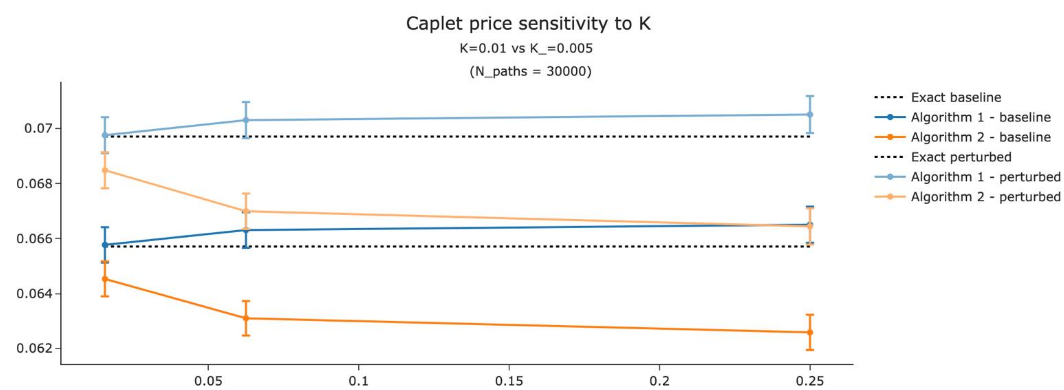
In the figure below, as the initial LIBOR rate L_0 decreases, the caplet price declines. This is expected, since a caplet functions as a barrier call option on the LIBOR rate—lowering L_0 reduces the likelihood of reaching the strike, especially under the presence of a barrier, thereby decreasing its value. Additionally, the variance observed in the simulation results also decreases with a lower L_0.



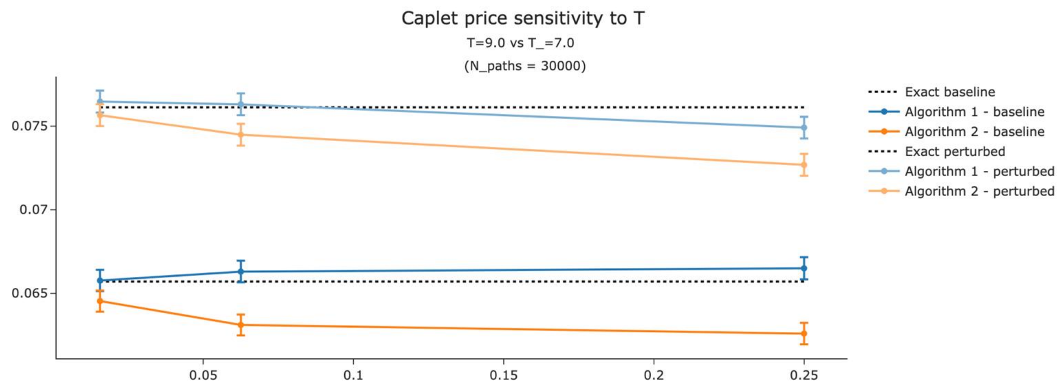
As the barrier level H decreases, the caplet price also decreases. This is expected, as we are dealing with a knock-out caplet—lowering H increases the likelihood of the barrier being hit, thereby nullifying the caplet and reducing its value. Additionally, the variance of the caplet payoff also decreases with a lower barrier.



An increase in volatility similarly leads to a lower caplet price, since higher volatility raises the probability of the barrier being breached, resulting in a knock-out.



Conversely, as the strike price K decreases, the caplet price increases. A lower strike enhances the intrinsic value of the caplet, making it more valuable.



Lastly, a shorter time to maturity results in a higher caplet price. With less time remaining, there is a reduced chance of hitting the barrier, which increases the likelihood that the caplet will remain active and thus more valuable.

4.3 Swaption

We compute the results for 10^6 simulations, using algorithm 1 and algorithm 2 respectively.

Table 3: Performance of Algorithm 6.1

h	**Error**	**Mean Exit Time**
0.25	$2.81 \times 10^{-2} \pm 1.60 \times 10^{-4}$	8.94
0.2	$2.46 \times 10^{-2} \pm 1.58 \times 10^{-4}$	9.04
0.125	$1.88 \times 10^{-2} \pm 1.56 \times 10^{-4}$	9.19
0.1	$1.66 \times 10^{-2} \pm 1.55 \times 10^{-4}$	9.24
0.0625	$1.29 \times 10^{-2} \pm 1.54 \times 10^{-4}$	9.32
0.03125	$8.72 \times 10^{-3} \pm 1.51 \times 10^{-4}$	9.41

Table 4: Performance of Algorithm 6.2

h	Error	Mean Exit Time
0.25	$2.53 \times 10^{-2} \pm 1.56 \times 10^{-4}$	8.99
0.2	$2.21 \times 10^{-2} \pm 1.54 \times 10^{-4}$	9.08
0.125	$1.70 \times 10^{-2} \pm 1.51 \times 10^{-4}$	9.20
0.1	$1.49 \times 10^{-2} \pm 1.50 \times 10^{-4}$	9.25
0.0625	$1.16 \times 10^{-2} \pm 1.48 \times 10^{-4}$	9.33
0.03125	$7.95 \times 10^{-3} \pm 1.45 \times 10^{-4}$	9.41

Note that in our simulations only about 8.8 % of paths actually triggered a rebound. With so few boundary corrections, both algorithms behave almost identically. As we refine the time step h, more trajectories will enter the "boundary layer" and undergo projection-and-rebound; in that regime, Algorithm 1 will clearly outperform Algorithm 2.

Likewise, our measured convergence rates (slopes) of approximately 0.55 reflect the fact that we have not yet reached the asymptotic regime. By pushing h still smaller—so that rebounds occur on, say, 30–50 % of paths—one should observe the slope rise toward 1.0 for Algorithm 1, while Algorithm 2 remains at 0.5.

Changing the parameters has well-understood effects on the payer knock-out swaption price:
- Number of resets N: Increasing N (more tenor dates after T_0) raises the annuity—i.e. the present-value sum of remaining coupon dates—and thus increases the option's value, all else equal. Reducing N shrinks the annuity and lowers the price.

- Option maturity T_0: A longer T_0 delays exercise, lengthens the accrual period, and typically enlarges the cumulative volatility of the swap rate, which tends to boost a vanilla swaption's value. However, under a knock-out, a longer window also raises the chance of breaching the barrier, an opposing effect that depends on the barrier level.

- Tenor interval δ: A larger δ reduces the total number of coupon payments for fixed T_0, cutting the annuity and lowering the premium. Conversely, a smaller δ increases the number of payments—and the option's sensitivity—and thereby raises the price.

- Strike K: A higher strike reduces the potential in-the-money payoff $\max(R-K,0)$ and thus depresses the option's value; lowering K has the opposite effect.

- Barrier level R_{up}: Moving R_{up} down brings the barrier closer to the current swap rate, sharply increasing knock-out probability and consequently reducing the premium. Raising the barrier protects the option longer and increases its value.

- Initial LIBOR curve L_0: Elevating all forward rates raises the initial swap rate R_0, increases the vanilla payoff probability, and typically raises the knock-out swaption's value—though it also slightly alters barrier-breach likelihood.

- Volatility σ: Greater σ widens the distribution of the swap rate at T_0, increasing both the chance of finishing in-the-money and the chance of early knock-out. The net effect on a knock-out option depends on the barrier's proximity to the initial rate: if the barrier is low, higher volatility can actually reduce value by triggering more early knock-outs.

- Correlation parameter β: Stronger correlation across tenors (small β) concentrates moves in the same direction, lowering the aggregate swap-rate volatility and thus reducing option value. Weaker correlation (large β) diversifies risk across tenors, boosting effective volatility and raising the price.

Overall, the knock-out swaption is most sensitive to the barrier level and the volatility, followed by the payment structure (N, δ), then by the initial curve, and finally by the correlation parameter.

4.4 Limits

Caplet
The second algorithm tends to underestimate the option price in most cases. However, it is fast, intuitive, and straightforward to implement, primarily because it avoids the use of a Brownian bridge. In contrast, the first algorithm relies on the Brownian bridge technique, which increases the computational cost slightly, but significantly reduces the bias, leading to more accurate pricing results.
Regarding the variance reduction technique, while it does indeed reduce the variance of the estimator, it comes at a considerable computational cost. This is especially true in the absence of a closed-form solution, where computing the gradient becomes necessary, making the method substantially slower.

Swaption
The swaption algorithm is inherently complex, as it involves solving a constrained optimization problem. The optimal solution is highly sensitive to the choice of the initial point, which is often nontrivial to determine. Additionally, relying on a uniform random variable to decide whether the algorithm should terminate introduces a degree of subjectivity. This can lead to a systematic underestimation of the option's price.

5 Conclusion

Overall, this study has been particularly insightful, largely due to the strong performance of the random walk algorithm. Notably, we successfully priced a barrier swaption while accounting for the correlation among LIBOR rates across different maturities — a feature that is challenging to implement using the Brownian bridge method. The results obtained in the caplet pricing section were also quite compelling. Moreover, we explored the trade-off between bias and variance: reducing bias typically results in increased variance. Therefore, selecting which to minimize should depend on the specific characteristics of the problem at hand. When comparing the random walk algorithm to the Brownian bridge method, we conclude that the former offers greater precision, albeit at the cost of increased complexity. The latter, while less accurate, remains easier to implement and more intuitive.