École polytechnique fédérale de Lausanne

Project 8

Stochastic Simulation

Stochastic approximation in mathematical finance

Authors:
Jean-Sébastien Delineau
Shivang Sachar

Professor:
Fabio Nobile
Assistant:
Matteo Raviola

1 European put option

1.1 Theory

Lemma 1.1. *Let X be a continuous random vector with probability density function f, and let g be a measurable function. Then,*

$$E[g(X)] = \int_{\mathbb{R}^m} g(x)f(x)dx. \tag{1}$$

Definition 1.2 (European Put option)**.** Let$\{S_t,\, t \in [0,T]\}$ be the GMB reflecting the stock price of an option with maturity $T > 0$. Then, the payout computed from the European put option is defined as a stochastic process:

$$\begin{aligned} f\colon [0,T] \times \Omega &\longrightarrow \mathbb{R} \\ (t,\omega) &\longmapsto \exp(-rT)(K - S_T(\omega))_+. \end{aligned} \tag{2}$$

As we will consider it as a non stochastic function which evaluates the random variable S_T, we will from now on use this definition:

$$\begin{aligned} f\colon \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto \exp(-rT)(K - x)_+. \end{aligned} \tag{3}$$

Where $(x)_+ = \max\{0, x\}$.

Definition 1.3 ((European) Option price)**.** The option price reflected of a GBM $\{S_t,\, t \in [0,T]\}$ evaluated with an European put option, is defined as:

$$I = E[f(S_T)] = \int_{\mathbb{R}} f(x)p(x)dx. \tag{4}$$

Where $p(x)$ is the pdf of the GBM, and the last equation follows from Lemma[1.1].

Theorem 1.4 (Black-Scholes formula)**.** *With the notation of the Definition [1.3], we obtain a closed form solution:*

$$I = \exp(-rT)K\Phi[w^*(\sigma)] - s_0\Phi[w^*(\sigma) - \sigma\sqrt{T}] \quad \textit{with } w^*(\sigma) = \frac{1}{\sigma\sqrt{T}}(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T). \tag{5}$$

Proof. Since S_t is a GBM using Lemma [**??**], we write $S_t \sim e^{log(s_0)+(r-\frac{\sigma^2}{2})t+\sqrt{\sigma^2 t}Y}$ where $Y \sim \mathcal{N}(0,1)$. Therefore using Lemma[1.1] we obtain:

$$E[f(S)] = \exp(-rT)E[(K - S_T)_+] = \frac{\exp(-rT)}{\sqrt{2\pi}}\sqrt{T}\int_{\mathbf{R}} \mathbb{1}_{\{K \geq S_T\}}(K - S_0\exp((r - \frac{\sigma^2}{2})T + \sigma x)\exp(\frac{-x^2}{2})dx$$

$$= \underbrace{K\sqrt{T}\frac{\exp(-rT)}{\sqrt{2\pi}}\int_{\mathbf{R}} \mathbb{1}_{\{K \geq S_T\}}\exp(\frac{-x^2}{2})dx}_{A} - \underbrace{S_0\sqrt{T}\frac{\exp(-rT)}{\sigma\sqrt{2\pi}}\int_{\mathbf{R}} \mathbb{1}_{\{K \geq S_T\}}\exp((r - \frac{\sigma^2}{2})T + \sigma x - \frac{x^2}{2})dx}_{B}.$$

Let $\mathbf{\Phi}$ denote the CDF of the standard normal distribution, as $\mathbb{1}_{\{K \geq S_T\}} = \mathbb{1}_{\{x \leq \frac{1}{\sigma\sqrt{T}}(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T)\}}$

$$= \underbrace{\exp(-rT)K\Phi[\frac{1}{\sigma\sqrt{T}}(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T)]}_{A} - \underbrace{S_0\Phi[\frac{1}{\sigma\sqrt{T}}(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T) - \sigma\sqrt{T}]}_{B}.$$

Therefore:

$$E[f(S)] = \underbrace{\exp(-rT)K\Phi[w^*(\sigma)]}_{A} - \underbrace{S_0\Phi[w^*(\sigma) - \sigma\sqrt{T}]}_{B} \quad \text{with } w^*(\sigma) = \frac{1}{\sigma\sqrt{T}}(\ln(\frac{K}{S_0}) - (r - \frac{\sigma^2}{2})T).$$

□

Theorem 1.5. *The option price as a function of the volatility $\sigma \longmapsto I(\sigma)$ is increasing, in particular:*

$$I \in [(e^{-rT}K - s_0)_+, e^{-rT}K]. \tag{6}$$

Proof. By the Black-Scholes formula, we have

$$I(\sigma) = e^{-rT}K\Phi[w^*(\sigma)] - S_0\Phi[w^*(\sigma) - \sigma\sqrt{T}]$$

where

$$w^*(\sigma) = \frac{1}{\sigma\sqrt{T}}\left(\ln\left(\frac{K}{S_0}\right) - \left(r - \frac{\sigma^2}{2}\right)T\right).$$

As $\sigma \to 0$, both $w^*(\sigma) \to \infty$ and $w^*(\sigma) - \sigma\sqrt{T} \to \infty$, so $I(\sigma) \to e^{-rT}K - S_0$.

As $\sigma \to \infty$, $w^*(\sigma) \to \infty$ and $w^*(\sigma) - \sigma\sqrt{T} \to 0$, so $I(\sigma) \to e^{-rT}K$. □

Remark 1.6. The previous Theorem yields that the following equation admits a unique solution for any $I_{market} \in [(e^{-rT}K - s_0)_+, e^{-rT}K]$:

$$I_{market} - I(\sigma) = 0 \tag{7}$$

In particular, any root finding algorithm applied on the previous equation to retrieve σ^* will converge easily. For any $\sigma \in [0, \sigma^*]$ we have $I(\sigma) \leq 0$ and for any $\sigma \in [\sigma^*, \infty)$ we have $I(\sigma) \geq 0$.

1.2 Results

1.2.1 Framework

Following Remark [1.6], we have decided to implement the Robin-Monro Algorithm in the following way:

$$\sigma_{n+1} = \sigma_n - \alpha_n \hat{J}(\sigma_n).$$

Where,

$$\hat{J}_N(\sigma_n) := \hat{I}_N(\sigma_n) - I_{market} = \frac{1}{N}\sum_{i=1}^{N}(Z^{(i)} - I_{market}) = \frac{1}{N}\sum_{i=1}^{N}\tilde{Z}^{(i)}.2$$

With $\tilde{Z}^{(i)} = Z^{(i)} - I_{market}$, and

$$\hat{I}_N(\sigma) = \frac{1}{N}\sum_{i=1}^{N} Z^{(i)}.$$

A Crude Monte Carlo approximation of the option price.

Remark 1.7. Assuming $\sigma_n \to \sigma^*$ the iterative scheme will converge with the Strong Law of Large Numbers.

Throughout the whole document we consider the following constant, interest rate $r = 5\%$, maturity time $T = 0.2$, initial asset price $S_0 = 100$.

1.3 Question: 1 & 2

Assuming strike price $K = 120$, and option price $I_{market} = 22$. By remark [1.6], there exists a unique solution σ^* to the equation $I(\sigma) - I_{market} = 0$. A bisection method applied to find the root of the equation gives the optimal value $\sigma_* = 0.508$.

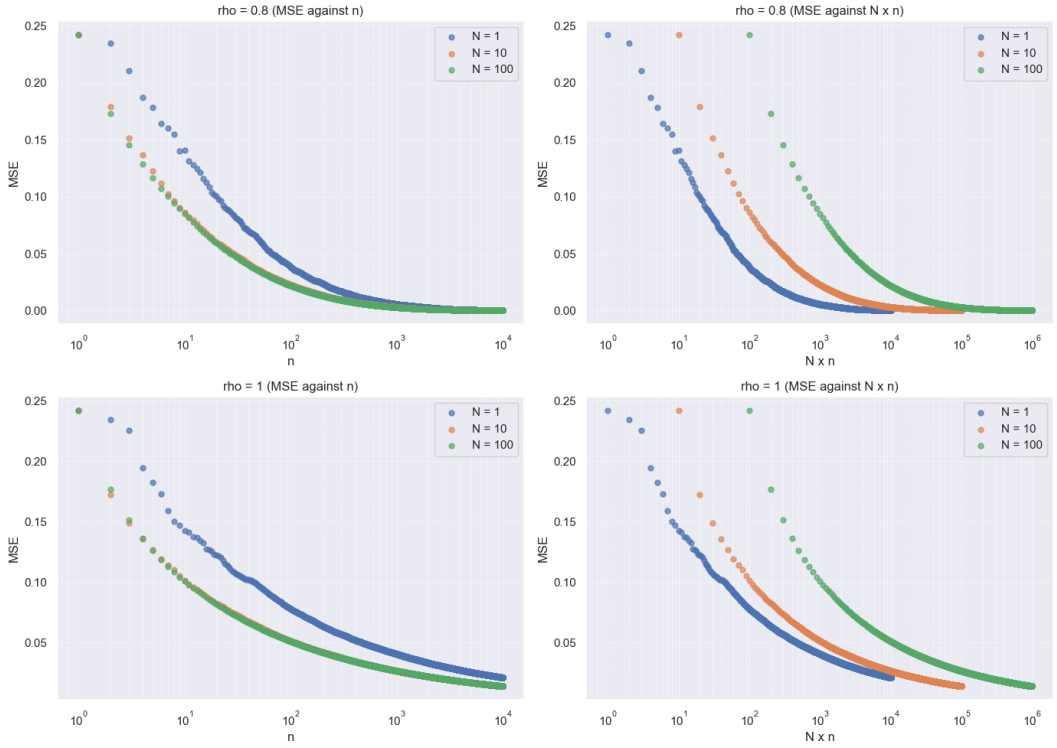

Figure 1: Convergence rate of MSE

We now implement the RM algorithm to find the implied volatility $\hat{\sigma}$ for $N \in \{1, 10, 100\}$ and $\rho \in \{0.8, 1\}$. To estimate the MSE $\mathbb{E}[(\hat{\sigma}_n - \sigma_*)^2]$), we repeat the simulation 500 times. The results are reported on Figure [1.

To calculate the convergence rate, we compute the slope of the linear fit of the log-log transformed data. If a linear model gives the equation

$$\log(\text{MSE}) = -\alpha \log(n) + \text{constant}$$

then α is the convergence rate. Implementing this in python gives the following results:

ρ	N	**Convergence Rate**
0.8	1	0.7487
0.8	10	0.7702
0.8	100	0.7844
1	1	0.2878
1	10	0.2899
1	100	0.2902

As we can see, the convergence rate is better when $\rho = 0.8$ and improves as N gets larger. This is no surprise, since the estimation $\hat{J}_N(\sigma)$ improves with larger values of N. However, the added benefit of the estimation $\hat{\sigma}$ is fairly minimal once $N \geq 10$, as depicted on the Figure [2].

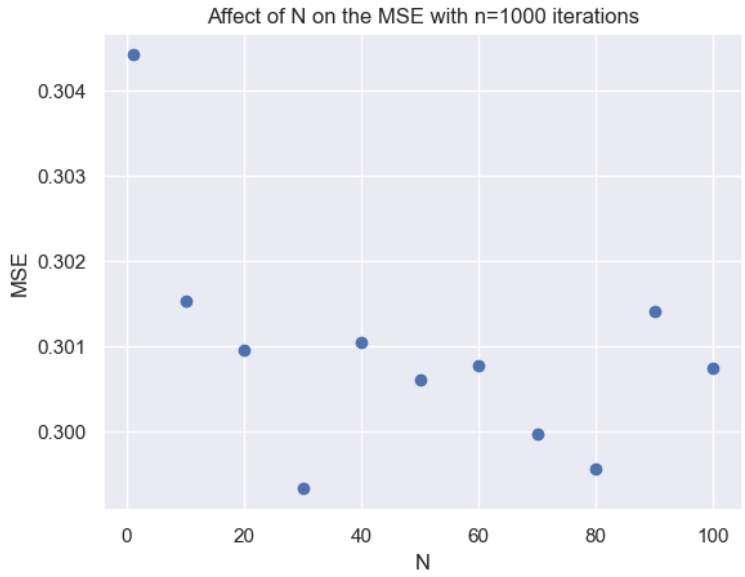

Figure 2: Dependence of MSE on N

2 Asian put option

2.1 Theory

As shown in Section [1], the European put option exclusively takes into account the final value of our stock price $\{S_t, t \in [0,T]\}$. In order to obtain a more accurate representation of the payoff or an alternative indicator thereof, there is a desire to formulate a function that specifies the payoff with the trajectory of the stock price as an input, which is precisely the objective achieved by the Asian option.

Remark 2.1. Throughout this section, we define $S = (S_{t_1}, \dots, S_{t_m})$ the random vector obtained from the geometric Brownian motion $\{S_t, t \in [0,T]\}$ evaluated at time $\{t_1, \cdots, t_m\}$, with $t_i = iT/m$.

Definition 2.2 (Asian Put Option)**.** For $\{S_t, t \in [0,T]\}$ GBM reflecting the stock price, the payout computed from the asian put option is defined as a stochastic process:

$$\begin{aligned} f\colon [0,T]^m \times \Omega^m &\longrightarrow \mathbb{R} \\ (t_1, \cdots, t_m, \omega) &\longmapsto \exp(-rT)(K - \overline{S}_T(\omega))_+ \quad \text{with} \quad \overline{S}_T(\omega) = \frac{1}{m}\sum_{i=1}^{m} S_{t_i}(\omega_i). \end{aligned} \tag{8}$$

As for the European put option we will consider it as a non stochastic function:

$$\begin{aligned} f\colon \mathbb{R}^m &\longrightarrow \mathbb{R} \\ (x_1, \cdots, x_m) &\longmapsto \exp(-rT)(K - \frac{1}{m}\sum_{i=1}^{m} x_i)_+. \end{aligned} \tag{9}$$

Definition 2.3 ((Asian) Option price)**.** The option price of this path dependant process S is defined as:

$$I = E[f(S)] = \int_{\mathbb{R}^m} f(S)p(S)ds \quad \text{p(S) the joint probability distribution of S.} \tag{10}$$

Remark 2.4. Generally, the option price obtained from an Asian put option does not admit a closed formula. Which is why we have to implement some Monte Carlo Method to get an approximation of its value. However, compared to the European put option in this instance we have to simulate an integral over m variable. One could then wonder if the random variable $\overline{S}_T = \frac{1}{m}\sum_{i=1}^{m} S_{t_i}$ follows a certain distribution. As it is an (averaged) sum of lognormal random variable. The short answer is no. There is no formula. Nevertheless, we can approximate its density using method as the Fenton-Wilkinson approximation [4], which perform poorly in the tails. [2].

Remark 2.5. Since we are considering an average summed of i.d.d random variables with finite mean and variance, the Central Limit Theorem or Berry-Essen theorem should apply (here we are considering marginally independent but for this argument we assume they are independent). As m increases, the density of the averaged sum should mimic the one of a normal density. But as the the log-normal distribution is heavy tailed and highly skewed this process is extremely slow [1].

As explained in the previous remarks, to compute the (Asian) option price we are forced to compute an integral over $\mathbb{R}^m$, which we will approximate with some Monte-Carlo Methods. First we will highlight properties of the joint distribution of S:

Remark 2.6. Since we will evaluate our stochastic process $\{S_t, t \in [0,T]\}$ at time increments $\{t_1, \ldots, t_m\}$. An alternative characterisation of it will be a discrete time continuous state space Markov chain. Therefore using conditional density and the memory-less property of the Markov chain,we have:

$$p(s_1, \ldots, s_m) = p_{S_m|S_{t_{m-1}} \ldots S_0}(s_m|s_{m-1}, \ldots, s_0) p_{S_{t_{m-1}}, \ldots, S_0}(s_{m-1}, \ldots, s_0)$$

$$= p_{S_m|S_{m-1}}(s_m|s_{m-1}) \ldots p_{S_1|S_0}(s_1|s_0).$$

Lemma 2.7. *Let $\{S_t, t \in [0,T]\}$ be a geometric Brownian motion that we evaluate at time $\{t_1, \cdots, t_n\}$ where $t_0 = 0$ and $S_0 = s_0$. Then,*

$$S_{t_j} = s_0 \exp(\sum_{i=1}^{j} X_{t_i}). \tag{11}$$

Proof. With the analogous characterization presented in the previous remark, we consider $(S_{t_1}, \ldots, S_{t_m})$ as a Markov chain. Using the defintion of a GBM we write: $S_{t_i} = S_{t_{i-1}} e^{X_{t_i}}$, i.e. the realization of the GBM at time t_{i-1} from t_i is the is the lognormal random variable $S_{i-1} e^{X_{t_i}}$ with $X_{t_i} \sim \mathcal{N}((r - \frac{\sigma^2}{2})t_i, \sigma^2 t_i)$. □

Therefore the joint density can be expressed as an independent product of lognormal where the starting value of the lognormal is the end value of the previous one:

Lemma 2.8. *Let $\{S_t, t \in [0,T]\}$ be a geometric Brownian motion that we evaluate at time $\{t_1, \cdots, t_n\}$ where $t_0 = 0$ and $S_0 = s_0$. Its joint density can be expressed as:*

$$p(x_1, \cdots, x_m) = \prod_{i=1}^{m} \frac{1}{x_i \sigma \sqrt{t_i - t_{i-1}}} \varphi(\frac{log[x_i/x_{i-1}] - (r - \sigma^2/2)(t_i - t_{i-1})}{\sigma \sqrt{t_i - t_{i-1}}}). \tag{12}$$

Where φ is the pdf of a standard Gaussian random variable.

Proof. Let us compute the pdf of a Geometric Brownian motion S_t starting at the value $s_0 \in \mathbb{R}$,

$$\mathbb{P}[S_t \leq x] = \mathbb{P}[(r - \sigma^2/2)t + \sigma W_{t-k} \leq \log(x/s_0)] = \mathbb{P}[W_t \leq \frac{1}{\sigma}(log(x/x_0) - (r - \sigma^2/2)t)]$$

$$= \mathbb{P}[X \leq \frac{1}{\sigma\sqrt{t}}(log(x/x_0) - (r - \sigma^2/2)t)] \quad X \sim \mathcal{N}(0,1).$$

$$= \Phi[\frac{(log(x/x_0) - (r - \sigma^2/2)t)}{\sigma\sqrt{t}}].$$

Therefore the pdf is given by:

$$p_{S_t}(x) = \frac{1}{x\sigma\sqrt{t}} \varphi[\frac{(log(x/x_0) - (r - \sigma^2/2)t)}{\sigma\sqrt{t}}]$$

Leveraging the result in Remark [2.6] we write:

$$p(s_1, \cdots, s_m) = \prod_{i=1}^{m} \frac{1}{s_i \sigma \sqrt{t_i - t_{i-1}}} \varphi\left(\frac{log[s_i/s_{i-1}] - (r - \sigma^2/2)(t_i - t_{i-1})}{\sigma \sqrt{t_i - t_{i-1}}}\right).$$

□

Theorem 2.9. *The option price as a function of the volatility $\sigma \longmapsto I(\sigma)$ is increasing, in particular:*

$$I \in [e^{-rT}(K - \frac{s_0}{m}\sum_{i=1}^{m} e^{irT/m})_+,\ e^{-rT}K]. \tag{13}$$

Proof. We could alternatively write the option price as:

$$I = E[\tilde{f}(X)] = E[\tilde{f}(x_1, \ldots, x_m)] = \int_{\mathbb{R}^m} \tilde{f}(x_1, \ldots, x_m)\tilde{p}(x_1, \ldots, x_m)dx_1, \ldots, x_m.$$

$$\tilde{f}(x_1, \ldots, x_m) = e^{-rT}(K - \underbrace{\frac{s_0}{m}\sum_{i=1}^{m} e^{(r-\sigma^2/2)iT/m+\sqrt{\sigma^2 T/m}\sum_{j=1}^{i} x_j}}_{A(\sigma)})_+.$$

With $\tilde{p}(x_1, \ldots, x_m)$ the joint probability distribution of X. where X is a standard Gaussian random vectors with covariance matrix $\Sigma = I_m$.

When $\sigma \to \infty$ $A(\sigma) \to 0$ as the $e^{-\sigma^2}$ win over the other terms, and when $\sigma \to \infty$ $A(\sigma) \to \frac{s_0}{m}\sum_{i=1}^{m} e^{irT/m}$. □

2.2 Results

In this part, we will use the same framework as explained 1.2.1, with constant $r = 5\%$, $S_0 = 100$, $T = 0.2$ and $m = 50$. As for the European option, in the algorithm we take $\alpha_0 = \frac{2}{K+S_0}$. We will denote by $(\sigma_n)_{n=1}^{\infty}$ the sequence of σ obtained by the algorithm, and $J_n := J(\sigma_n)$ the sequence of estimated value of J at each iterations.

2.2.1 Question 5: Crude Monte Carlo

Let $K = 120$ and $I_{market} = 22$, Theorem [2.9] yields that I is in the interval $(19.23, 118.81)$ for any value of σ.

The ρ parameter quantifies the aggressiveness of the algorithm, i.e. how big can be the difference of two consecutive values of the sequence σ_n. For this reason the α_0 taken, to obtain a fast converging algorithm we need $\rho = 0.5$ as depicted on the plot 3. However, this has a negative aspect as the values of σ_n will have more amplitude of oscillation around σ^* resulting in more variance of J_n around 0 when the algorithm converges. We took this decision as fast computing algorithm is always preferred.

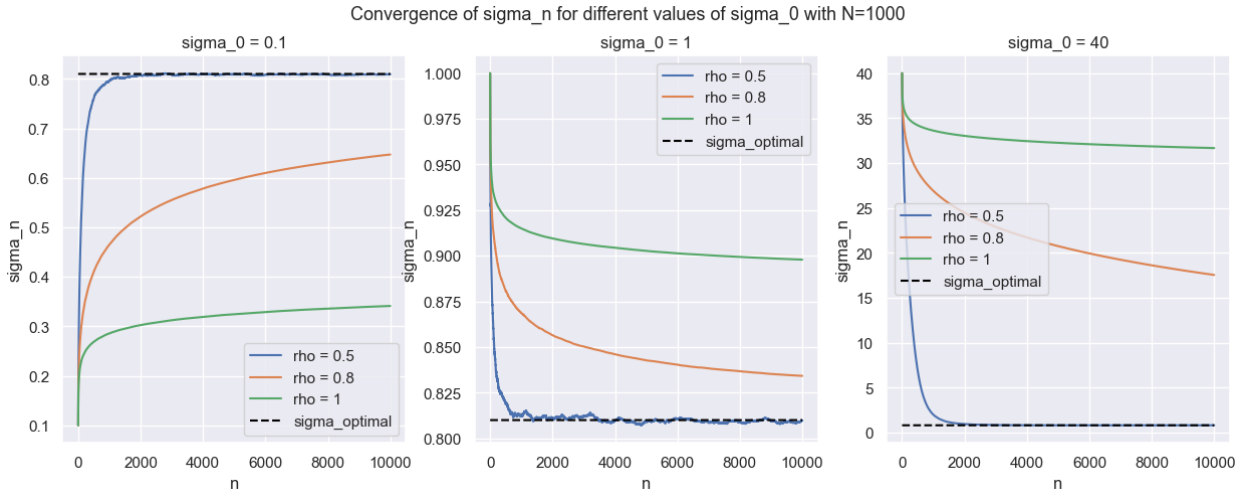

Figure 3: Aggressiveness of the algorithm

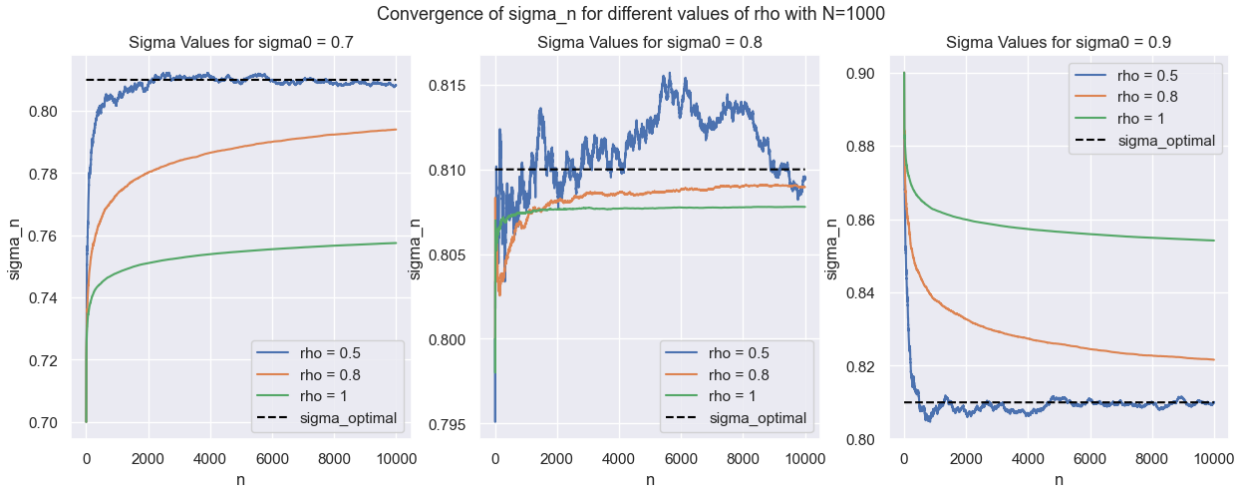

Figure 4: Step size

When implementing a stopping criteria, as the value of σ^* is unknown, the condition for stopping the algorithm must rely on the estimator J. Relying on the SLLN obtained from the CMC algorithm, if $\sigma_n \to \sigma^*$, $J_n \to 0$. Assuming $\sigma_0 > \sigma^*$ Theorem [2.9] yields that the first values of J_n will be positive, which will make σ_n decreases for these first values, till a point where it will undershoot the optimal value of σ^* and then oscillate around it with smaller and smaller differences in between consecutive values as shown in the middle plot of Figure 4. Experiments have shown that for these constants with $\sigma_0 < 15$ the sequence σ_n will stay in an interval of length one of the optimal value σ^* in less than 1000 iterations with $\rho = 0.5$. For example, consider Figure [3]. And if we were in a more volatile frame with $15 < \sigma_0 < 100$ it will only take 7000 iterations.

Therefore a more efficient way to implement the algorithm, would be to burn-in the first values till the estimator J changes sign with $\rho = 0.5$ and then continue with our ρ of choice. Which is the way we have decided to implement the algorithm. Our stopping condition is defined as the moving average of the last 1000 iterations of J_n to be smaller than a tolerance t_1 combined with end value $|J_n| < t_2$. There is no added benefit to take $t_2 < 10^{-2}$ as J_n oscillate randomly around 0. With this bound for t_2, our algorithm struggled to find a solution when $t_1 < 10e^{-3}$ due to the jumps size induced by $\rho = 0.5$. The following table summarize our findings for $\rho = 0.5$:

Statistic	**Mean**	**Standard deviation**
σ_n	0.81021	0.00107
Iterations	15945.45	11704.99317

Table 1: Monte Carlo Simulation of 20 samples

If a more precise solution is desired, one should either increases the value of ρ, but it will consequently increase the number of iterations needed as we can see on Figure [4]. Another way to obtain a more precise solution is to reduce the variance intrinsic to the generation of $\hat{J}$, which is what we do in the next part:

2.2.2 Question 6: Importance Sampling

To implement the Importance Sampling Algorithm we introduce an auxiliary pdf g(x) such that its curve mimics the one of the product of $f(x)$ and $p(x)$. Our candidate will belong in the the family of pdf introduced in Lemma [2.8] with a modified drift $\tilde{r}$. We will denote it $\tilde{p}(x) := p_{\tilde{r}}(x)$.
We will show that in dimension 1 (m=1), the optimal value $\tilde{r}$ depends on the starting point of our path S_0. To keep an efficient GBM generating algorithm, we will compute a common $\tilde{r}$ for each path $S = (S_{t_1}, \ldots, S_{t_m})$. The easiest and most efficient way to implement it for a dimension $m > 1$ with constant time increments, would be to assume that each paths S_{t_i} starts at the mean value of S_0 and S_T. As $E[S_t] = S_0 e^{rT}$, we obtain $\tilde{S}_0 = S_0(e^{rT}+1)/2 \approx 100.5$. Therefore we just need to find $\tilde{r}$ for the 1 dimensional case with starting point $\tilde{S}_0$.
We would like to find the pdf $\tilde{p}(x)$ mimicking the best the pdf $f(x)p(X)/I$, where I is the constant defined in Definition [2.3]. $\tilde{p}(x)$ should vanish outside of the Supp$(f^2(x)p^2(x))$ while mimicking the distribution. Therefore our conditions for optimization to retrieve optimal $\tilde{r}$, could be to have less probability possible where $f(x)p(x) \approx 0$ and having the closest first k moments.

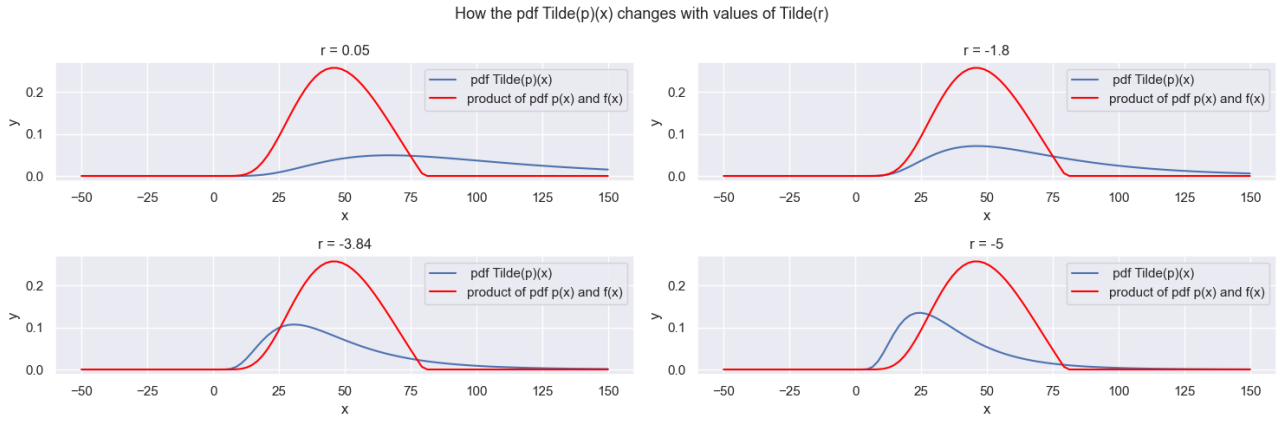

Figure 5: Influence of $\tilde{r}$ with K=80, $\sigma = 1.2$

As the distribution is highly skewed the latter is not the best criteria, nor retrieving $\tilde{r}$ to share the same unique stationary point, which is a global maximal as seen on the top plot of Figure [5] as too much mass is over K=80. Therefore we implemented the first criteria mentioned, which is to take the smallest $\tilde{r}$ to have 90% of the distribution under K. As the pdf is highly skewed, taking a threshold bigger than 90% will not yield a good result.

Figure [5] demonstrates how the distribution of $\tilde{p}(x)$ is modified when r is changed. On the top left we have the distribution with initial r. Top right is with $\tilde{r} = -1.8$ which makes the max of the distribution coincide. The bottom left is the optimal one from our algorithm and the bottom right is when $\tilde{r}$ overshoots. σ is defined as the σ associated to the first time $\hat{J}_n$ changes value.

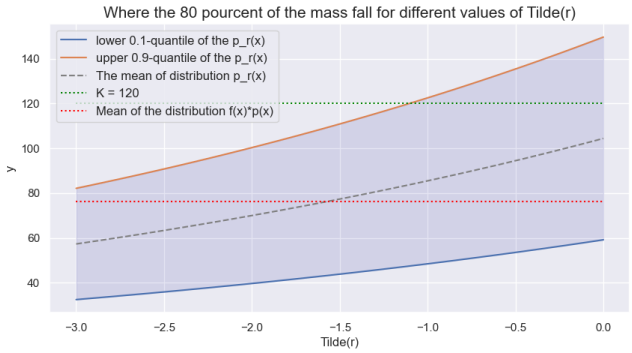

Figure 6: Influence of $\tilde{r}$ with $K = 120, \sigma^* = 0.81$

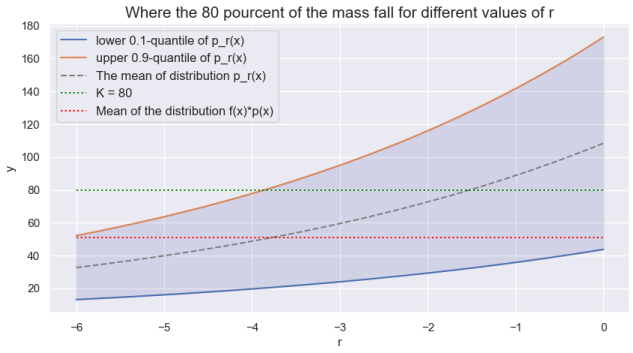

Figure 7: Influence of $\tilde{r}$ with $K = 80, \sigma = 1.2$

As explained before, with $\rho = 0.5$ in less than 1000 iterations in low volatility setting we are able to obtain a good approximation of σ^*. Therefore our algorithm is independent of our first guess σ_0. which is why we took σ_0 as the average of 10 σ_n each respective to the first $J_n(\sigma)$ changes sign for the first time. From the arguments of before, σ should stay close to this value, which is why we compute $\tilde{r}$ in this setting. $\tilde{r}$ is taken as the intersection of the curve of the 0.9-quantile and the constant curve K as shown on the Figures [6],[7]. Taking a higher α quantile will not yield good results as quantile curves grows to exponentially.

The fixed $\tilde{r}$ algorithm yields that the optimal value of $\tilde{r} = -3.8$ in this setting. If the algorithm is correctly implemented, it should provide significant variance reduction in the Monte Carlo approximation of $\hat{J}(\sigma_n)$. Since importance sampling is a variance reduction technique, we expect the algorithm to significantly outperform the standard RM algorithm when N is small. For our purposes, we shall take $N = 10$ and $\rho = 0.6$.

2.2.3 Question 7: (Sophisticated) Importance Sampling

Let $K = 80$, by Theorem [2.9], we have $I_{market} \in (0.0, 79.204)$ for any values of σ. Let us find the volatility σ^* solving equation [**??**] for $I_{market} = 3$.

We also implement the more sophisticated importance sampling strategy where the value of $\tilde{r}$ is chosen to be optimal depending on the current value σ_n.

$$\tilde{r}(\sigma) \approx \arg\min_{\eta} \frac{1}{\overline{N}} \sum_{i=1}^{\overline{N}} \frac{f^2(S^{(i)}; r, \sigma) p(S^{(i)}; r, \sigma)}{p(S^{(i)}; \eta, \sigma)}$$

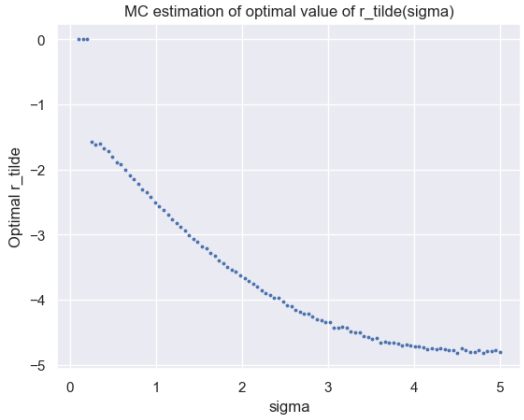

Figure 8: Optimal $\tilde{r}$ as a function of sigma, K=80,$\overline{N} = 10^5$

We shall now compare the performance of the three algorithms. To measure this, we run the three algorithms 20 times until the stopping criteria of point 5 is met and measure the average number of iterations n until the algorithm terminates, the average value of the final estimation σ_n and the standard deviation in the estimation.

Algorithm	**Average num. of iterations** n	**Mean value of** σ_n	**Standard deviation in** σ_n
Standard RM	31632	1.11100	0.00352
RM with IS	12983	1.10952	0.00192
RM with IS and optimal $\tilde{r}$	8080	1.10958	0.00295

This confirms that the RM algorithm with importance sampling and optimal choice of $\tilde{r}$ does indeed outperform the other two algorithms. In order to understand the effect of the parameter N on the convergence of the three algorithms, we plot a single run of the three RM algorithms for $N = 1, 10, 100$, until $n = 10^4$.

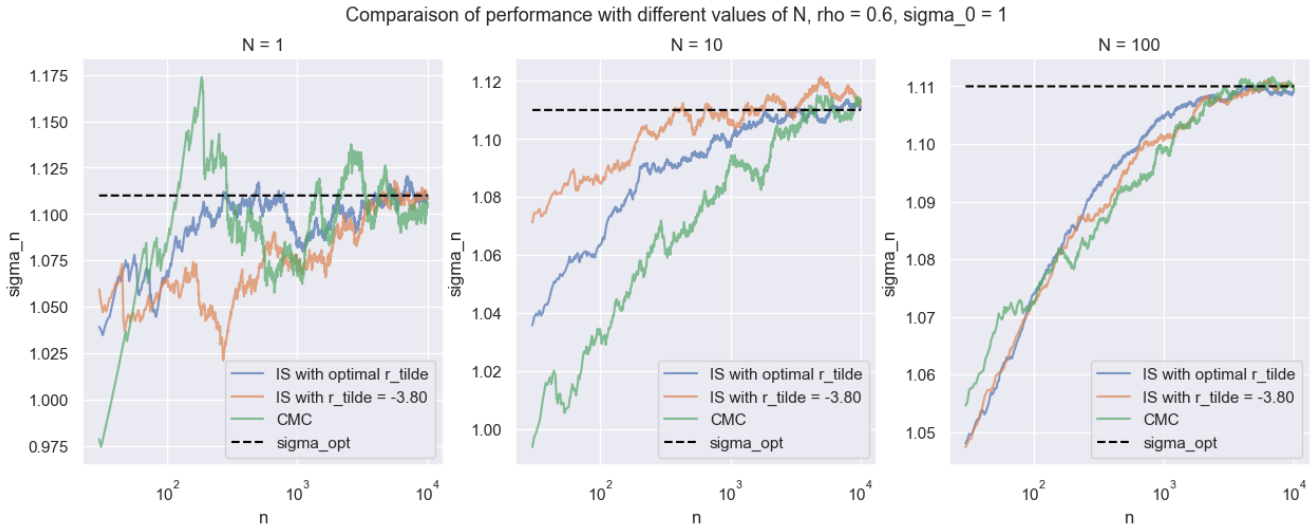

Figure 9: Reduced jump size of σ_n with (sophisticated) IS

As we can see, the blue curve (RM algorithm with importance sampling and optimal $\tilde{r}$) has less variance than both the green and orange curves, for all three values of N. To more concretely measure the variance, we take a window of size 5000 and plot the variance of the curve as the window scans over the values of σ_n.

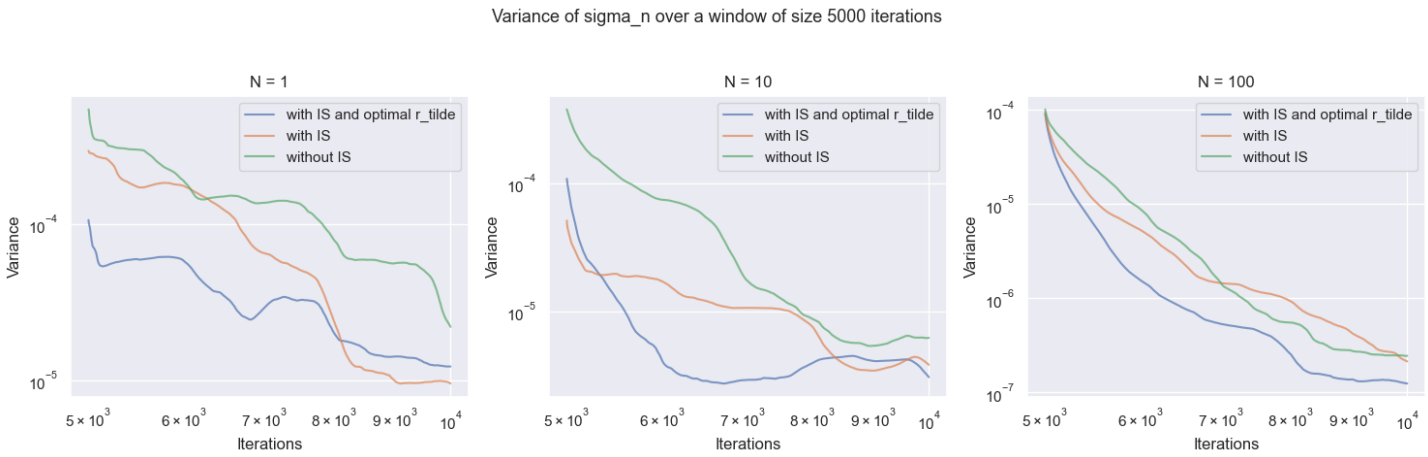

Figure 10: (Sophisticated) Imporance sampling Variance reduction

The above plot again verifies that the RM algorithm with importance sampling and optimal $\tilde{r}$ has less variance in its estimation of σ_* than the RM algorithm with importance sampling and fixed $\tilde{r} = -3.8$, which in turn has less variance than the standard RM algorithm.

3 Appendix - Code Snippets

We include the main functions that were integral to our project.

Listing 1: Simulation of a Geometric Brownian Motion

```
def Simulate_Stock_Price(S_0,sigma, r, T, m,N):
    """This function returns N GBM, each evaluated at time T/m,...,T with a starting value of S_0

    Parameters
    ----------
    S_0 : float
        Initial value of the geometric Brownian motion
    sigma : float
        Volatility of the geometric Brownian motion
    r : float
        Interest rate of the geometric Brownian motion
    T : float
        Time horizon of the geometric Brownian motion
    m : int
        Number of time steps
    N : int
        Number of GBM to generate

    Returns
    -------
    S : np.ndarray
        N GBM evaluated at time T/m,...,T (N_features, m_features)
    """
    simulations = np.ndarray((N, m))
    expXt = st.lognorm.rvs(s=np.sqrt(sigma ** 2 * T / m), loc=0, scale=np.exp((r - 0.5 * sigma ...
        ** 2) * T / m), size=m*N)
    expXt_reshaped = expXt.reshape((N, m), order='F')  # Reshape the array to m rows and N columns
    simulations = np.cumprod(np.concatenate([np.ones((N, 1)) * S_0, expXt_reshaped], axis=1), ...
        axis=1)
    return simulations[:, 1:]
```

Listing 2: RM Algorithm for an Asian Put Option with a Stopping Criterion

```
def RM_Asian_with_Stopping(N, rho, K, S0, T, r, I_market, m,sigma_0,Max_iteration = ...
    10**6,window_size=1000,tol1=10**(-3),tol2=10**(-2)):
    """This function returns the sequence sigma_n and J_n for the option price to equal to ...
        market price using a variant of the Robbins-Monro algorithm
    Parameters
    ----------
    n : int
        Number of iterations
    N : int
        Number of MonteCarlo simulation to approximate J
    rho : float
        Agressivness of the algorithm
    K : float
        Strike price of the put option
    S0 : float
        Initial value of the geometric Brownian motion
```

```
    T : float
        Time horizon of the geometric Brownian motion
    r : float
        Interest rate of the geometric Brownian motion
    I_market : float
        Market price of the option
    m : int
        Number of time steps GBM
    sigma_0 : float
        Initial guess of the volatility
    Returns
    -------
    sigma_estim : np.ndarray
        Estimated volatility at each iteration  (n_features, )
    mean_it : np.ndarray
        Estimated valued of J at each iteration  (n_features, )
    """

    if rho > 0.5:
        sigma_0,i = Sign_changing(K, S0, T, r, I_market, m, sigma_0)
    else:
        i=1

    alpha_0 = 2/(K+S0)
    sigma_cur = sigma_0
    sigma_estim = deque([sigma_0])
    J_estim = deque([])

    while i < Max_iteration:
        alpha_n = alpha_0 / i ** rho

        # Calculate Jhat
        simulations = Simulate_Stock_Price(S0, sigma_cur, r, T, m, N)
        avg_stock_prices = np.mean(simulations, axis=1)
        Z = np.exp(-r*T) * np.maximum(K - avg_stock_prices, 0) - I_market
        Jhat, est_std = np.mean(Z), np.std(Z)
        sigma_cur = sigma_cur - alpha_n * Jhat
        sigma_estim.append(sigma_cur)
        J_estim.append(Jhat)

        # Stopping Criterion
        if i == window_size:
            mean_value = np.mean(J_estim)
        if i > window_size:
            removed_value = J_estim.popleft()
            mean_value = mean_value + (Jhat - removed_value) / window_size
            if abs(mean_value) < tol1 and abs(Jhat) < tol2:
                break
        i += 1

    return sigma_estim,i
```

Listing 3: Finding the best value of a fixed $\tilde{r}$ for Importance Sampling (Question 6)

```
def function_r_tilde(K, S0, T, r, I_market, m, sigma_0,M=10):
    """ Initial guess of sigma_* with sign_changing + MC of 10 values obtained + output ...
        0-9-quantile"""
    H = np.zeros(shape=(M,2))

    for k in range(M):
        sigma, i = Sign_changing(K, S0, T, r, I_market, m, sigma_0)
        H[k,0] = sigma
        H[k,1] = i

    sigma_suboptimal = np.mean(H[:,0])
    # Mean of the path
    mu_path = (1+np.exp(r*T) )/2 * S0

    def intersection(x):
        return st.lognorm.ppf(0.90, scale = np.exp(np.log(mu_path) + (x-0.5*sigma_suboptimal**2) ...
            * T), s = np.sqrt( sigma_suboptimal ** 2 * T))-K

    r_tilde = bisect(lambda sigma: intersection(sigma), -10, 10)

    return r_tilde
```

Listing 4: Finding the optimal value of $\tilde{r}(\sigma)$ for Importance Sampling (Question 7)

```
def get_r_opt(sigmas, S0, K, r, T, m, N):
    def g(eta, sigma, simulations):
        likelihood_ratios = w(simulations[:, -1], S0, sigma, r, eta, T)

        avg_stock_prices = np.mean(simulations, axis=1)
        payoffs_squared = (np.exp(-r*T) * np.maximum(K - avg_stock_prices, 0))**2
        variance = np.mean(np.multiply(payoffs_squared, likelihood_ratios))
        return variance

    result = np.empty((len(sigmas),))
    for i, sigma in enumerate(sigmas):
        simulations = Simulate_Stock_Price(S0, sigma, r, T, m, N)
        result[i] = minimize_scalar(g, args=(sigma,simulations)).x
```

Listing 5: RM Algorithm with Importance Sampling and optimal $\tilde{r}$

```
def RM_Asian_with_IS_opt(n, N, rho, K, S0, T, r, I_market, m, sigma_0, r_opt):
    sigma_0 = 1
    alpha_0 = 2/(K+S0)

    sigma_estim = np.empty((n,))
    sigma_cur = sigma_0
    sigma_estim[0] = sigma_cur
    for i in range(1,n):
        alpha_n = alpha_0 / i ** rho

        # Calculate Jhat with IS
        r_tilde = r_opt(sigma_cur)
        simulations = Simulate_Stock_Price(S0, sigma_cur, r_tilde, T, m, N)
        likelihood_ratios = w(simulations[:, -1], S0, sigma_cur, r, r_tilde, T)
        avg_stock_prices = np.mean(simulations, axis=1)
        payoffs = np.exp(-r*T) * np.maximum(K - avg_stock_prices, 0)
        Jhat = np.mean(np.multiply(payoffs, likelihood_ratios)) - I_market

        sigma_cur = sigma_cur - alpha_n * Jhat
        sigma_estim[i] = sigma_cur

    return sigma_estim
```

References

[1] Andrey Akinshin. "Central limit theorem and log-normal distribution". In: (2023). URL: https://aakinshin.net/posts/clt-lnorm/.

[2] Daniel Dufresne. "SUMS OF LOGNORMALS". In: (). URL: https://www.soa.org/globalassets/assets/files/static-pages/research/arch/2009/arch-2009-iss1-dufresne.pdf.

[3] Damir Filipovic and Elena Perazzi. "Probability and Stochastic Calculus Continuous Time Lecture notes". In: (2023). URL: https://moodle.epfl.ch/pluginfile.php/3298104/mod_resource/content/0/stocalc_20231106_lectures_8-14.pdf.

[4] Marwane Ben Hcine1 and Ridha Bouallegue. "On the Approximation of the Sum of Lognormals by a Log Skew Normal Distribution". In: (). URL: https://arxiv.org/ftp/arxiv/papers/1502/1502.03619.pdf%20https://aakinshin.net/posts/clt-lnorm/.

[5] Fabio Nobile. "Stochastic Simulation". In: (2023).

[6] Jerome Le Ny. "Introduction to Stochastic Approximation Algorithms". In: *NY Course* (2009). URL: https://www.professeurs.polymtl.ca/jerome.le-ny/teaching/DP_fall09/notes/lec11_SA.pdf.