Exposure profile enhancement using cashflow information

September 2024

Abstract

We've introduced 2 methods to interpolate present trade values, which serves to both reduce simulation costs and increase the granularity of the exposure profile in Monte-Carlo simulations. The first method is less computationally efficient since it requires to simulate a brownian bridge while the second method takes advantage of the fact that we are in a monte carlo simulation and we are going to take the expectation of the simulations so the interpolation has to be good on average.
Two models are introduced using the secnod method : sqrt-linear and a lognormal-normal mixture model.
Those techniques are specifically designed to capture the inherent volatility of present values which cannot be achieved via linear interpolation, while also accurately reflecting sudden trade jumps within the exposure profile. By doing so, we attempt to decrease computation time without compromising the accuracy of the results, as evidenced by maintaining a similar RMSE in the XVA calculations. This balance between efficiency and precision is crucial in ensuring the reliability of the model while optimizing performance. The precision measurement approach will be based on the MSE of XVA values, which will be discussed in detail. Performance measurement will be evaluated by the computation time and memory usage.
Those methods have to be integrated to a multilevel monte carlo framework. We recall how MLMC works and we describe how this integration is done.

The structure of the thesis will be :

1. General statements and definition of the problem
2. Propositions of solutions
3. Integration of the solutions to the MLMC framework
4. Results

Aknowledgments

I would like to express my sincere gratitude to my supervisor, Jon Linkins, for his guidance and support throughout this work. I also extend my thanks to Benjamin Piau and Ben Burnett for their insightful discussions and assistance.

Contents

1 Introduction

From a bank's perspective, XVA computations are notoriously expensive, largely because they rely on extensive Monte Carlo simulations that demand significant computational resources. The conventional approach to managing these costs involves sacrificing precision, which inevitably leads to a loss of crucial signal information. However, this trade-off between cost and accuracy is far from ideal.

Our goal is to develop a solution that reduces the overall computation time while preserving the integrity of the simulated signals. Rather than compromising on precision, we aim to maintain the accuracy of the most critical components of the calculation. By doing so, we seek to strike a balance where the essential details of the exposure profile are retained, ensuring that the results remain robust and reliable, even as we optimize the efficiency of the computational process. This approach not only addresses the cost concerns but also enhances the quality of the XVA computations, aligning with the bank's need for both precision and performance.

2 General setup

2.1 XVA example : CVA

The concept of XVA (eXpected Value Adjustments) refers to a family of valuation adjustments applied to the pricing of derivative contracts to account for various risk factors. These adjustments include Credit Valuation Adjustment (CVA), Debit Valuation Adjustment (DVA), Funding Valuation Adjustment (FVA), among others. The purpose of XVA is to incorporate the costs associated with counterparty credit risk, funding costs, capital requirements, and other risks into the valuation of derivatives. The XVA value can be described as a sum of different value adjustments depending on the risks taken and the existing models. It's important to note that, although XVAs are applied to the price of individual products at the time of trading, each of the XVA components (CVA, DVA, ...) are calculated daily for the entire portfolio of trades with a counterparty. In our case, we only have interest in this end of the day risk assesment calculation where all trades are considered together. More details on XVA can be found in [3] and [6].

CVA, or Credit Valuation Adjustment, is one of the most prominent components of XVA. It represents the expected loss due to the counterparty's potential default. Specifically, CVA is the difference between the risk-free value of a derivative and its true value considering the possibility of counterparty default. It is crucial for financial institutions to account for CVA to ensure accurate pricing and risk management of derivatives, reflecting the counterparty credit risk embedded in these contracts.

The formula for CVA is given by ([6]):

$$\text{CVA} = (1-R)\int_0^T \mathbb{E}^Q\left[\frac{B_0}{B_t}E(t) \mid \tau = t\right] d\text{PD}(0,t)$$

where:

- R is the recovery rate, the proportion of value recovered in the event of default.
- $\mathbb{E}^Q[\cdot]$ denotes the risk-neutral expectation, used to calculate expected values under the risk-neutral measure.
- B_0 and B_t are the discount factors at times 0 and t, respectively, used to discount cash flows to present value.
- $E(t)$ is the exposure at time t, representing the potential loss in the event of default. It is determined by the positive difference between the trade's present value (PV) and the collateral posted. Specifically, the exposure is given by
$$E(t) = \max(\text{PV}(t) - \text{Collateral}(t), 0),$$
which reflects the risk that the collateral does not fully cover the PV of the trade.

- τ is the time of default, the random time at which the counterparty defaults.
- $\mathrm{PD}(0,t)$ is the cumulative probability of default from time 0 to time t, representing the likelihood that the counterparty defaults by time t.
- T is the maturity of the contract, the time at which the derivative contract ends.

2.2 Computation method

In order to compute the CVA, we will use the Monte Carlo technique.

Knowing that default probabilities are a separate subject not covered here, the steps are as follows:

1. Define a simulation schedule for the trade's lifecycle.
2. Simulate paths for the trade's future values over time.
3. Calculate the pathwise exposure values and average them accross the simulated paths to obtain the exposure profile.
4. Integrate this exposure profile over the defined schedule against the probability of default.

This approach highlights that the future values of portfolio trades are the cornerstone of any XVA computation. They provide the foundation for applying a Monte Carlo approach to generate exposure profiles, which are essential for calculating various XVA metrics.

In general we can estimate that the overall cost of an XVA calculation using the Monte-Carlo technique will be a multiple of the number of paths times the number of dates. The idea behind interpolating PV values is that the interpolation is way faster than pricing a trade.

2.3 Our problem

Recalling the CVA formula, we note that it requires integrating the expected exposure over a continuous time schedule. In practice, however, this schedule must be discretized for simulation purposes. Banks, dealing with numerous counterparties and performing extensive calculations of this nature, face constraints in both time and computational resources (e.g., RAM). As a result, the time schedule is often set to a low frequency (e.g., one simulation point per month instead of daily). This low-frequency discretization, combined with the cashflow jumps in the exposure profile, introduces an integration bias when numerically integrating the expected exposure.

This picture will allow us to visualize what that means.

Figure 1: Expected exposure with simulation dates and missed spikes

In this example we see that the low frequency simulation schedule is responsible for a large part of the error. This example has been chosen to see what a smooth exposure profile looks like but in reality, most counterparties have really discontinuous profiles with high punctual cashflows. This is an example which make us realize how crucial it is to capture them, the exposure is almost zero everywhere except on one single thin spike. Missing it in the schedule would make us completely misscalculate the XVAs.

Figure 2: This is the figure legend for the missing image.

2.4 HJM 1 factor for interest rates swap pricing

We need to understand that the pricing model we use is crucial for our analysis. The way we model trades determines their price dynamics and, as a consequence, the exposure profiles shape.

It is also important to recognize that each counterparty may have multiple types of trades, each with its own pricing model.

Since most of a bank's trades are interest rate swaps, I will briefly explain the model used, which will help clarify the assumptions made later.

The Heath-Jarrow-Morton (HJM) framework is a widely used approach for modeling the evolution of interest rates. In the context of a one-factor HJM model, the dynamics of the instantaneous forward rate, $f(t,T)$, are described by the following stochastic differential equation (SDE) [9]:

$$df(t,T) = \alpha(t,T)\,dt + \sigma(t,T)\,dW_t$$

where:

- $\alpha(t,T)$ is the drift term, ensuring no arbitrage,
- $\sigma(t,T)$ is the volatility term,
- W_t is a standard Brownian motion.

The drift term $\alpha(t,T)$ is typically derived from the volatility structure to ensure the absence of arbitrage. Given this, the forward rate dynamics are predominantly driven by the stochastic term involving $\sigma(t, T)$ and dW_t, which implies that over small intervals, $df(t, T)$ is normally distributed.

The price of a zero-coupon bond, $P(t, T)$, maturing at time T can be expressed as:

$$P(t, T) = \exp\left(-\int_t^T f(t, u)\, du\right)$$

It is important to note that $P(t, T)$ is log-normally distributed due to the normally distributed integral of $f(t, u)$.

For a plain vanilla interest rate swap, the present value (PV) of the fixed leg, PV_{fixed}, is given by:

$$PV_{\text{fixed}} = \sum_{i=1}^{N} C \cdot P(0, T_i)$$

where C is the fixed payment amount and $P(0, T_i)$ are the discount factors at times T_i.

The present value of the floating leg, PV_{float}, is given by:

$$PV_{\text{float}} = \sum_{i=1}^{N} \left(L_{T_{i-1}}(T_i - T_{i-1})\right) P(0, T_i)$$

where $L_{T_{i-1}}$ is the forward rate set at time T_{i-1} and $T_i - T_{i-1}$ is the day count fration.

The price of the swap itself, V_{swap}, is the difference between the present values of the floating and fixed legs:

$$V_{\text{swap}} = PV_{\text{float}} - PV_{\text{fixed}}$$

The dynamics of the swap pricing can be complex due to the integration of stochastic processes. However, it can be approximated:

The PV of the fixed and floating legs are sums of several terms involving discount factors and forward rates, which are derived from the stochastic forward rates $f(t, T)$. Given that these factors can be approximated as log-normal, their sum (the swap price) can be approximated as normal.

This point will be discussed in more detail in section 4.3.

3 Litterature review

The problem originated from the topic of margin period of risk, in fact [2] treats how to take margin period of risk into account when computing exposure profiles. Margin Period of Risk (MPOR) is the time period during which a party to a financial contract is exposed to risk due to market movements or counterparty default, before collateral can be liquidated or adjusted to cover the exposure.

The project then deviated to the optimisation of PV production and interpolation methods still inspired by [2] setup and especially the section 8.2 of his paper where he uses a brownian bridge to extrapolate trades present values.
An interpolation using a brownian bridge method will be presented but since our objective here is to interpolate PVs instead of simulating them, we can start from this setup and find alternatives like done in [8]. In order to not simulate, we want an interpolation working in expectation and this is done in a simple example in [8] when the author wants to compute E[Wt2 — Wt1, Wt3] with t1 ¡ t2 ¡ t3 and a brownian motion Wt. The lognormal-normal mixture interpolation model was inspired by [4] where it is not used for interpolation and not used in a Monte-Carlo setup but we can adapt it to our purpose.
Since the XVA monte carlo framework is running Multilevel monte carlo ([5], [7]), the integration of the interpolation method will be done in that framework but we need to figure out how. We get from [1] a method to estimate the MSE of the XVA calculation in the MLMC setup.

4 Methodology

4.1 Preliminary statements

From the perspective of the XVA desk, it becomes necessary to make certain approximations to handle this problem effectively. In practice, the simulations employ various pricing models, each tailored to a specific financial product. As a result, each product category may exhibit distinct distributions and behaviors.

For example, consider two different product categories: equity derivatives and credit derivatives. Equity derivatives might follow a lognormal distribution, which assumes that the underlying asset prices are always positive and exhibit proportional changes. On the other hand, credit derivatives could follow a heavy-tailed distribution like the Student's t-distribution, which accounts for the potential of extreme events or credit defaults. These differing distributions impact how risks are modeled and priced, which makes our task more complicated since we work from a counterparty perspective.

4.2 Requirements : Discounting, Jumps, Volatility

There are three main components in PV dynamics, and we will describe each of them to understand how they manifest in the mathematical model later.

1. **Discounting/Drift**: To get an accurate approximation of PV points, it is important to note that the PVs could be undiscounted and would behave as martingales when discounted.

2. **Jumps**: These jumps refer to the discontinuities caused by the cashflows during the trade's life. The trade value will approximately jump by the cashflow amount when it occurs.

3. **Volatility**: In this context, volatility refers to the fact that present values (PVs) have non-deterministic behavior due to the stochastic nature of the underlying factors such as interest rates, equities, and other market variables. Consequently, PVs, as a dynamic process, inherit volatility from these stochastic drivers. The problem would be simpler if linear interpolation were sufficient to handle PV dynamics. However, it fails to reproduce the volatility that real PVs exhibit, making the inclusion of volatility a necessary component of the model.

Figure 3: Expected exposure with linearly interpolated PVs

4.3 The normal/log-normal assumption

The selection of a single model for all PV types isn't feasible because each trade type has its own PV distribution, such as normal, lognormal, or even more complex mixtures.

We recall that the majority of a bank's trades are IRSwaps, whose PVs are distributed as sums of lognormals. In [10], the author establishes a theorem that helps explain the distribution of the sum of lognormal variables when they are positively correlated. However, there is a conflict due to the correlation of each leg in the PV price: if the legs are highly correlated, the theorem can be applied; if not, we cannot say much, and the best approximation is likely a normal distribution.

Since the normal model has shown good results for our purpose and is computationally efficient, it is the model we are using for interpolation. However, a mixture model would provide a better approximation, though its downside is the need to estimate its parameters.

4.4 Methods preliminary

The methods have been separated in two different sections because their essence differ.

- The Joint distribution interpolation (Brownian bridge)
- The Marginal distribution interpolation (sqrt-linear and log-linear)

The first method is called "joint distribution" interpolation because for each point M_t that we would like to interpolate between two simulated points M_{ti} and $M_{t_{i+1}}$, we would like to get a good joint distribution $(M_t, M_{t_{i+1}})$.

The second interpolation however is called "marginal distribution" interpolation because we want to get a good distribution of M_t only.

4.5 Method 1 : Joint distribution interpolation, brownian bridge interpolation

In this section, we consider a scenario where the volatility σ_s is deterministic but not necessarily constant over the interval $[t_i, t_{i+1}]$. Our objective is to interpolate the value of a stochastic process M_t between two time points t_i and t_{i+1}, taking into account this variable volatility.

We assume for readability reasons that there are no cashflows and the mark to market prices are already discounted. The next section will account for cashflows and how to integrate them to this kind of interpolation scheme.

We recall from [2] that the expected exposure will be of the form :

$$\mathbb{E}[(M_{t_{i+1}} - \phi(M_t))^+]$$

But in a scenario where the CSA is strong and the collateral call is instant (no MPOR) then $\phi = Id$ and $\mathbb{E}[(M_{t_{i+1}} - \phi(M_t))^+] = \mathbb{E}[(M_{t_{i+1}} - M_t)^+]$ wich make us want a good joint distribution of $(M_t, M_{t_{i+1}})$.

The process M_t is assumed to follow the stochastic differential equation (SDE):

$$M_t - M_{t_i} = \int_{t_i}^{t} \sigma_s \, dW_s,$$

where W_t denotes a standard Brownian motion. Under this assumption, the increment $M_t - M_{t_i}$ is normally distributed with a mean of zero and a variance given by the integral:

$$\mathrm{Var}(M_t - M_{t_i}) = \int_{t_i}^{t} \sigma_s^2 \, ds.$$

This expression results from the fundamental property of stochastic integrals, where the variance of an integral with respect to Brownian motion is the integral of the square of the integrand. Hence, the variance of the process increment between t_i and t directly depends on the temporal evolution of σ_s.

Next, we determine the conditional expectation of M_t, given the values at times t_i and t_{i+1}. Due to the martingale property of the process M_t, this expectation is a linear interpolation between M_{t_i} and $M_{t_{i+1}}$:

$$\mathbb{E}[M_t \mid M_{t_i}, M_{t_{i+1}}] = M_{t_i} + \frac{\int_{t_i}^{t} \sigma_s^2 \, ds}{\int_{t_i}^{t_{i+1}} \sigma_s^2 \, ds} \left(M_{t_{i+1}} - M_{t_i} \right).$$

This linear interpolation is justified by the fact that, in a martingale framework, the best estimate in the least squares sense is given by a linear interpolation between the known values at the endpoints of the interval.

The conditional variance of M_t, given M_{t_i} and $M_{t_{i+1}}$, can be expressed by considering the deterministic nature of σ_s. The conditional variance is then:

$$\operatorname{Var}(M_t \mid M_{t_i}, M_{t_{i+1}}) = \int_{t_i}^{t} \sigma_s^2 \, ds \cdot \frac{t_{i+1} - t}{t_{i+1} - t_i}.$$

This conditional variance is obtained by subtracting the variance associated with the linear interpolation from the total variance over the interval $[t_i, t]$. The term $\frac{t_{i+1}-t}{t_{i+1}-t_i}$ arises from the covariance structure of the Brownian increments, ensuring that the conditional variance correctly accounts for the interpolation between M_{t_i} and $M_{t_{i+1}}$.

By combining the components of conditional expectation and conditional variance, we can express the interpolated process M_t as follows, using both deterministic and stochastic elements:

$$M_t = \mathbb{E}[M_t \mid M_{t_i}, M_{t_{i+1}}] + \sqrt{\int_{t_i}^{t} \sigma_s^2 \, ds \cdot \frac{t_{i+1} - t}{t_{i+1} - t_i}} \cdot Z,$$

where $Z \sim \mathcal{N}(0, 1)$ is a standard normal random variable independent from the brownian motion used for the simulation of M_{t_i} and $M_{t_{i+1}}$.

Thus, the final interpolation formula becomes:

$$M_t = M_{t_i} + \frac{\int_{t_i}^{t} \sigma_s^2 \, ds}{\int_{t_i}^{t_{i+1}} \sigma_s^2 \, ds} (M_{t_{i+1}} - M_{t_i}) + \sqrt{\int_{t_i}^{t} \sigma_s^2 \, ds \cdot \frac{t_{i+1} - t}{t_{i+1} - t_i}} \cdot Z.$$

This interpolation formula captures both the linear interpolation in expectation and the stochastic fluctuation in variance over the interval $[t_i, t_{i+1}]$, while incorporating a deterministic yet variable σ_s. By combining the deterministic and stochastic terms, we obtain an accurate estimate of M_t that faithfully reflects the dynamic characteristics of the process within this interval.

4.6 Method 2 - Marginal distribution interpolation, sqrt-linear interpolation

Given a martingale stochastic process M_t, we aim to interpolate M_t for $t_i < t < t_{i+1}$ using the known values at the endpoints M_{t_i} and $M_{t_{i+1}}$.

We assume that σ_s is a deterministic function, and W_s is a standard Brownian motion with independent increments and normally distributed changes.

The process is defined by the relationship:

$$M_t - M_{t_i} = \int_{t_i}^{t} \sigma_s \, dW_s,$$

where W_s is a Brownian motion.

To find an expression for M_t that interpolates between M_{t_i} and $M_{t_{i+1}}$, we start by normalizing the increment of the process:

$$\frac{M_t - M_{t_i}}{\sqrt{\int_{t_i}^{t} \sigma_s^2 \, ds}} \overset{law}{=} \frac{M_{t_{i+1}} - M_{t_i}}{\sqrt{\int_{t_i}^{t_{i+1}} \sigma_s^2 \, ds}}.$$

This equality indicates that the normalized increments of M_t over $[t_i, t]$ and $[t_i, t_{i+1}]$ share the same distribution.

Under the assumption that σ_s is deterministic, the integrals don't cause problems in our computation and can be kept as :

$$\int_{t_i}^{t} \sigma_s^2 \, ds \quad \text{and} \quad \int_{t_i}^{t_{i+1}} \sigma_s^2 \, ds.$$

For now, we retain these integrals in their general form.

From the relationship above, the interpolated value of M_t can be expressed as:

$$M_t - M_{t_i} = \sqrt{\frac{\int_{t_i}^{t} \sigma_s^2 \, ds}{\int_{t_i}^{t_{i+1}} \sigma_s^2 \, ds}} (M_{t_{i+1}} - M_{t_i}).$$

If we assume $\sigma_s = \sigma$ is constant over the interval $[t_i, t_{i+1}]$, the integrals simplify to:

$$\int_{t_i}^{t} \sigma^2 \, ds = \sigma^2 (t - t_i),$$

$$\int_{t_i}^{t_{i+1}} \sigma^2 \, ds = \sigma^2 (t_{i+1} - t_i).$$

Substituting these into the interpolation formula, we obtain:

$$M_t - M_{t_i} = \sqrt{\frac{t - t_i}{t_{i+1} - t_i}} (M_{t_{i+1}} - M_{t_i}).$$

Therefore, the interpolated value of M_t for $t_i < t < t_{i+1}$ is given by:

$$M_t = M_{t_i} + \sqrt{\frac{t - t_i}{t_{i+1} - t_i}} (M_{t_{i+1}} - M_{t_i}).$$

This formula directly provides the interpolated value of M_t based on the known values M_{t_i} and $M_{t_{i+1}}$ without the need for additional random variables. Since we tried to get a good approximation of the marginal law of M_t, this approach can be used when calculating an expectation depending solely on the marginal distribution of M_t and no joint distribution.

4.7 Method 3 - Marginal distribution interpolation, lognormal-normal mixture interpolation

As discussed above, a we cannot find a perfect interpolation for every trade as their PV distribution are different.
In this part, we assume the distributions of the PVs we want to interpolate can be approximated by a mixture of a log-normal and a normal. For this, we write the SDE :

$$dM_t = ((1 - |\gamma|)M_t + \gamma X_0)(\mu dt + \sigma dW_t)$$

where

- W_t is a Brownian motion.
- $-1 \leq \gamma \leq 1$ determines the degree of mixture of the two processes.
- X_0 is a constant that can be considered a volatility factor.

As we work with discounted PVs, we suppose M_t is a martingale and we can write the equation this way :

$$dM_t = ((1 - |\gamma|)M_t + \gamma X_0)\sigma dW_t$$

Let's manipulate the SDE :

$$dM_t = d(M_t + \frac{\gamma M_0}{1 - |\gamma|}) = (M_t + \frac{\gamma M_0}{1 - |\gamma|})(\sigma(1 - |\gamma|))dW_t$$

We can now define :

- $\sigma_n = \gamma M_0 \sigma$
- $\sigma_{ln} = (1 - |\gamma|)\sigma$
- $a = \frac{\sigma_n}{\sigma_{ln}} = \frac{\gamma M_0}{1-|\gamma|}$

The SDE can now be transformed into

$$d(M_t + a) = (M_t + a)\sigma_{ln} dW_t$$

This equation simply defines a log-normal diffusion process with no drift and a displacing parameter a. It is important to note that a is defined as a ratio of the normal and log-normal standard deviations.

- In the limit as $\gamma \to 0$ ($\Leftrightarrow a \to 0$), the process X_t reduces to a pure log-normal process.
- In the limit as $|\gamma| \to 1$ ($\Leftrightarrow a \to \infty$), the process X_t reduces to a pure normal process.

The solution of the SDE is:

$$X_t + a = (X_0 + a) \exp\left[-\frac{1}{2}\sigma_{ln}^2 t + \sigma_{ln} W_t \right]$$

In a context where two values M_{t_i} and $M_{t_{i+1}}$ are known at times t_i and t_{i+1}, and one wishes to interpolate the value of M_t for $t_i < t < t_{i+1}$, we can rewrite this as:

$$X_t + a = (X_{t_i} + a) \exp\left[-\frac{1}{2}\sigma_{ln}^2 (t - t_i) + \sigma_{ln}(W_t - W_{t_i}) \right]$$

We can use the following law equality:

$$\frac{\ln\left(\frac{M_t + a}{M_{t_i} + a}\right) + \frac{1}{2}\sigma_{ln}^2 (t - t_i)}{\sigma_{ln}\sqrt{t - t_i}} \quad \overset{\mathcal{L}}{=} \quad \frac{\ln\left(\frac{M_{t_{i+1}} + a}{M_{t_i} + a}\right) + \frac{1}{2}\sigma_{ln}^2 (t_{i+1} - t_i)}{\sigma_{ln}\sqrt{t_{i+1} - t_i}} \sim \mathcal{N}(0,1)$$

This allows us to write an interpolation formula for M_t in terms of the known values M_{t_i}, $M_{t_{i+1}}$, and the constant σ_{ln}:

$$M_t + a = (M_{t_i} + a) \left(\frac{M_{t_{i+1}} + a}{M_{t_i} + a} \right)^{\sqrt{\omega(t, t_i, t_{i+1})}} \lambda(t, t_i, t_{i+1}, \sigma_{ln})$$

Where:

- $\lambda(t, t_i, t_{i+1}, \sigma_{ln}) = \exp\left[\frac{1}{2}\int_t^{t_{i+1}} \sigma_{ln,s}^2 ds\right] = \exp\left[\frac{1}{2}\sigma_{ln}^2 (t_{i+1} - t)\right]$ is the Ito term that we didn't have in the normal model.
- $\omega(t, t_i, t_{i+1}) = \frac{t - t_i}{t_{i+1} - t_i}$ is the time ratio factor, which simplifies because σ_{ln} is constant.

Isolating M_t, we obtain the final expression for the interpolation:

$$M_t = (M_{t_i} + a) \left(\frac{M_{t_{i+1}} + a}{M_{t_i} + a} \right)^{\sqrt{\omega(t, t_i, t_{i+1})}} \lambda(t, t_i, t_{i+1}, \sigma_{ln}) - a$$

This approach tries to ensure that the interpolation of the process, even in the presence of potentially negative values, remains consistent and avoids any violation of the positivity constraint of the log-normal process.

An interesting thing to do at this point is to compute the Taylor expansion of

But we know that:

$$f_{y_0,y_1,\omega}(a) = (y_0+a)\left(\frac{y_1+a}{y_0+a}\right)^{\sqrt{\omega}} \lambda(\sigma_{ln}) - a$$

We have:

$$\begin{aligned}
f_{y_0,y_1,\omega}(a) &= \frac{(y_1+a)^{\sqrt{\omega}}}{(y_0+a)^{\sqrt{\omega}-1}}\lambda(\sigma_{ln}) - a \\
&= a\left[\left(1+\frac{y_1}{a}\right)^{\sqrt{\omega}}\left(1+\frac{y_0}{a}\right)^{1-\sqrt{\omega}}\lambda(\sigma_{ln}) - 1\right] \\
&= a\left[\left(1+\frac{y_1\sqrt{\omega}}{a} + \mathcal{O}_{a\to\infty}\left(\frac{1}{a^2}\right)\right)\left(1+\frac{(1-\sqrt{\omega})y_0}{a} + \mathcal{O}_{a\to\infty}\left(\frac{1}{a^2}\right)\right)\lambda(\sigma_{ln}) - 1\right] \\
&= a\left[\left(1+\frac{y_1\sqrt{\omega}+(1-\sqrt{\omega})y_0}{a} + \mathcal{O}_{a\to\infty}\left(\frac{1}{a^2}\right)\right)\lambda(\sigma_{ln}) - 1\right] \\
&= a\left[\left(1+\frac{y_1\sqrt{\omega}+(1-\sqrt{\omega})y_0}{a}\right)\lambda(\sigma_{ln}) + \mathcal{O}_{a\to\infty}\left(\frac{1}{a^2}\right) - 1\right] \\
&= a(\lambda(\sigma_{ln})-1) + y_1\sqrt{\omega}\lambda(\sigma_{ln}) + (1-\sqrt{\omega})y_0\lambda(\sigma_{ln}) + \mathcal{O}\left(\frac{1}{a}\right) \\
&= a(\lambda(\sigma_{ln})-1) + \lambda(\sigma_{ln})(y_0+\sqrt{\omega}(y_1-y_0)) + \mathcal{O}_{a\to\infty}\left(\frac{1}{a}\right)
\end{aligned}$$

On the other hand, let's analyze $\lambda(\sigma_{ln})$. We have:
We start with the expression for $\lambda(\sigma_{ln})$:

$$\lambda = \exp\left(\frac{1}{2}\sigma_{ln}^2(t_{i+1}-t)\right) = \exp\left(\frac{1}{2}\sigma_{ln}^2(t_{i+1}-t)(1-|\gamma|)\sigma\right)$$

Given that $a = \frac{\gamma M_0}{1-|\gamma|}$, we have:

$$|\gamma| = \frac{|a|}{|M_0|+|a|} = \frac{1}{1+\left|\frac{M_0}{a}\right|} = 1 - \left|\frac{M_0}{a}\right| + o\left(\frac{1}{a^2}\right)$$

Using this approximation, we can rewrite $\lambda(\sigma_{ln})$ as:

$$\lambda(\sigma_{ln}) = \exp\left(\frac{1}{2}\sigma_{ln}^2(t_{i+1}-t)\right) = \exp\left(\frac{1}{2}\left(\left|\frac{M_0}{a}\right| + o\left(\frac{1}{a^2}\right)\right)^2\sigma^2(t_{i+1}-t)\right)$$

Expanding the exponential function, we get:

$$\lambda(\sigma_{ln}) = 1 + \frac{1}{2}\left|\frac{M_0}{a}\right|^2\sigma^2(t_{i+1}-t) + o\left(\frac{1}{a^4}\right)$$

We can now write the full development:

$$f_{y_0,y_1,\omega}(a) = (y_0 + \sqrt{\omega}(y_1 - y_0)) + \frac{1}{2}\left|\frac{M_0^2}{a}\right|\sigma^2(t_{i+1} - t) + \mathcal{O}_{a\to\infty}\left(\frac{1}{a}\right)$$

The zeroth-order term corresponds exactly to the result given by the sqrt-linear interpolation. This means if $a \gg M_{t_i}$ and $a \gg M_{t_{i+1}}$ then calibrating any parameter (a or σ_{ln}) is useless since they don't appear anymore, we should just use the sqrt-linear interpolation.

In order to use the interpolation, we need to calibrate two parameters per time frame, this is discussed in the following paragraph.

Parameter estimation

If we take the SDE we just solved and restrict it to the interval $[t_i, t_{i+1}]$, we can assume that σ_{ln} is given by $\sigma_{ln,i}$, where σ_{ln} is modeled as a step function. Similarly, the other parameters, γ and a, are also step functions, remaining constant on the interval $[t_i, t_{i+1}]$.

With this approach, we generate a grid of values based on the Monte Carlo path-dates grid. For each time frame $[t_i, t_{i+1}]$, we have N paths and two parameters to estimate: γ and $\sigma_{ln,i}$.

For this purpose, we are going to use the maximum likelihood estimator. This can be done because we have the following relationship:

$$Y_i^j = \frac{\ln\left(\frac{M^j_{t_{i+1}}+a_i}{M^j_{t_i}+a_i}\right) + \frac{1}{2}(\sigma_{ln,i})^2(t_{i+1} - t_i)}{\sigma_{ln,i}\sqrt{t_{i+1} - t_i}} \sim \mathcal{N}(0,1)$$

where $1 \leq j \leq N$ is the path index.

Using the log likelihood function of i.i.d. Gaussian variables since the paths are assumed to be independent:

$$\ell(\sigma_{ln,i}, a_i; Y_i^1, \ldots, Y_i^N) = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\sum_{j=1}^{N}(Y_i^j)^2$$

We take from [4] that the folowing estimator of $\sigma_{ln,i}$ can be obtained :

$$\hat{\sigma}(a_i) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\ln\left(\frac{M^j_{t_{i+1}} + a_i}{M^j_{t_i} + a_i}\right)\right)^2}$$

Then, we can take $a_i = \arg\max\left(\ell(\sigma_{ln,i}(a_i), a_i; Y_i^1, \ldots, Y_i^N)\right)$ numerically and compute the corresponding numerical value for $\sigma_{ln,i}$.

The estimators are:

- $\hat{a_i} = \arg\max\left(\ell(\sigma_{ln,i}(a_i), a_i)\right)$
- $\hat{\sigma_{ln,i}} = \hat{\sigma}(\hat{a_i})$

The philosophy here is to estimate a_i and $\sigma_{ln,i}$ for every time frame using the Monte Carlo paths and a maximum likelihood estimator (MLE). Once we have the numerical values for these parameters, we can use them for interpolation.

An important point to note is that we could first assume $\sigma_{ln,i}$ is a step function for estimation purposes, and then, when we need it for interpolation, treat it as piecewise affine. In other words, after estimating $\sigma_{ln,i}$ and $\sigma_{ln,i+1}$, we apply the interpolation formula for t using the integrals of σ_{ln} in $\lambda(t, t_i, t_{i+1}, \sigma_{ln})$. To compute the integral, we can employ the trapezoidal rule to approximate the value accurately.

The advantage of this model is that it is quite convenient when we do not know the exact nature of the trade we are interpolating. It also serves as a good approximation for interest rate swaps (IRSwaps) since a sum of correlated lognormals can be approximated by a single lognormal, according to [10].

However, the downside of this technique is that it adds computation time in determining the parameters. Moreover, even with good parameter estimates, the model remains an approximation, where the square-root-linear model already provides a reasonable approximation even for lognormal distributions. Therefore, the potential precision gain from this technique might not justify the additional computational cost.

4.8 Adding cashflow jumps

The way we introduce cashflow jumps can be explained as follows:

We are in the time interval $[t_i, t_{i+1}]$, where we know M_{t_i} and $M_{t_{i+1}}$, and we would like to interpolate for any $t_i \leq t \leq t_{i+1}$. There are discounted cashflows $C_{t_{i,k}}$ with $t_i \leq t_{i,k} \leq t_{i+1}$ and $1 \leq k \leq n$. We introduce a function:

$$\Pi(t_i, t) := \sum_{t_{i,k} \leq t} C_{t_{i,k}}.$$

The method involves removing the sum of the discounted cashflows in the coefficient part of the interpolation and then subtracting the current sum of the discounted cashflows back as we interpolate. This results in the following formula, using the square-root linear interpolation as an example:

$$M_t = M_{t_i} + \sqrt{\frac{t - t_i}{t_{i+1} - t_i}} \left(M_{t_{i+1}} - (M_{t_i} - \Pi(t_i, t_{i+1})) \right) - \Pi(t_i, t),$$

This approach ensures that jumps occur at the correct time, based on the last term, while the rest of the interpolation proceeds as if there were no jumps, avoiding any conflict with previous calculations.

Implementation optimisations

We can observe that computationally, for each $i, M_{t_{i+1}} - (M_{t_i} - \Pi(t_i, t_{i+1}))$ is a constant. This implies that we do not need to recompute Π every time, and the value can be stored for any point we might interpolate within the interval $[t_i, t_{i+1}]$.

$\Pi(t_i, t)$, which represents the sum of the cashflows up to time t, is simply an accumulator that sums the cashflows if we have passed their respective dates within the interpolation interval.

4.8.1 Example

Imagine a scenario where we have four time points: t_0, t_1, t_2, t_3, and t_4.

- We are given the values of M_{t_0} and M_{t_4}.
- A (discounted) cashflow C arrives at time t_2.
- Our goal is to interpolate M_t at time t_1 and t_3, considering the cashflow information.

To properly interpolate M_{t_1} and M_{t_3} while accounting for the cashflow at t_2, we follow the steps outlined below:

1. **Collect the total cashflow information over the interval $[t_0, t_4]$.** The cashflow C occurring at time t_2 needs to be taken into account in the interpolation. This cashflow is considered part of the evolution of M_t between t_0 and t_4.
2. **Adjust the formula used for interpolation by subtracting the cashflow from the non-starting point part of the interpolation.** The interpolation should be adjusted by subtracting the cashflow from the formula that determines the evolution of M_t between t_0 and t_4. This ensures that the cashflow information is properly embedded in the process.
3. **Perform the interpolation, applying cashflow corrections at the appropriate time.** The interpolation can be done using the sqrt-linear interpolation formula as follows:

$$M_{t_1} = M_{t_0} + \sqrt{\frac{t_1 - t_0}{t_4 - t_0}} \left(M_{t_4} - (M_{t_0} - C)\right),$$

and

$$M_{t_3} = M_{t_0} + \sqrt{\frac{t_3 - t_0}{t_4 - t_0}} \left(M_{t_4} - (M_{t_0} - C)\right) - C,$$

This formula ensures that the process evolves smoothly from M_{t_0} to M_{t_4} while accounting for the cashflow C at t_2.

4.9 Discounting

The discounting problem can be treated separately from the interpolation itself. We considered our PVs were already martingales so we can interpolate them directly and output a discounted result.

In the implementation the PVs were undiscounted and the output needed to be undiscounted also. The trick we can use is just to discount them (as well as the cashflows), interpolate them and then undiscount the result.

Other problematics are specific to the XVA desk : trades can be in different currencies, using different discount curves, but at some point everything needs to be in one currency for aggregation given that the Monte Carlo simulation happens under a unique measure, thus currency. Those considerations require extra care in practive from an implementation perspective but we leave it out from this thesis as this is deviating from our main subject and is really library specific.

4.10 The interpolation schedule

There is a baseline schedule used for asset pricing, this is normally the schedule used for integrating/aggregating the XVA payoff. In general this schedule has two patterns :

- Has regular dates aligned with key market dates, such as credit instrument tenors, since the CVA will be hedged by taking positions on credit instruments.
- Is denser in certain areas, particularly at the start of the trade's lifetime, due to assumptions about the cashflow profile, which suggests most cashflows will occur close together early on.

We are now considering adding an other layer of dates outside the baseline schedule which will be the interpolation schedule.

The choice of the interpolation schedule over the trade's lifetime is left to the user's creativity. The main options are:

- Uniform schedule
- Uniform schedule, more dense at the beginning
- Interpolate right before and right after cashflow dates
- Dense interpolation in areas with dense cashflows

In our case, we choose the uniform schedule since it is easier to work with alongside the rest of the library. This approach ensures that the simulated points and interpolated points are interlaced together, and we do not favor certain areas over others, regardless of where the cashflows occur.

Additionally, if we need to downsample an expected exposure, the process is straightforward in the uniform case, whereas it can be more challenging in other modes.

4.11 Comparison of sqrt-linear to linear

This section aims at graphically seeing the differences between our interpolation modes. We are in a setup where we have two simulated points and we would like to see what is the shape difference of our interpolations compared to linear. This gives us a feeling of how the volatility is taken into account.

Figure 4: Increasing PV example

Figure 5: Decreasing PV example

4.12 Intermediate results : Granularity

Lets have a counterparty with many different kinds of trades and let's apply a uniform interpolation schedule. We use the sqrt-linear interpolation here.

Figure 6: Expected exposure with sqrt-linear interpolated PVs

We observe that the interpolated points allow us to better capture jumps and reduce the likelihood of missing spikes. This should lead to a lower integration bias if our goal is accuracy, rather than reducing the computational time.

However, if our objective is to maintain similar results while achieving faster computations, the results presented in the following sections will illustrate this more effectively.

5 MLMC for XVA calculation

MLMC is a framework, we can and intent to implement the interpolation with and without the framework layer. This means our methods can be used in classic Monte Carlo without MLMC.

5.1 MLMC

Multilevel Monte Carlo (MLMC) ([5], [7]) is a variance reduction technique that enhances the efficiency of standard Monte Carlo methods, particularly when dealing with complex stochastic systems such as those driven by stochastic differential equations (SDEs). The method exploits the trade-off between computational cost and accuracy by using a hierarchy of simulations at different levels of resolution. This section details the theoretical foundation and practical implementation of MLMC, along with the mathematical formulations.

Standard Monte Carlo Method

In the standard Monte Carlo method, we are typically interested in estimating the expected value of some quantity P, which may depend on a stochastic process. For example, P could represent the payoff of a financial derivative or a physical quantity modeled by an SDE. The Monte Carlo estimate is given by:

$$\mathbb{E}[P] \approx \hat{P}_N = \frac{1}{N} \sum_{i=1}^{N} P^{(i)},$$

where $P^{(i)}$ is the outcome from the i-th sample, and N is the total number of independent samples.

The variance of this estimator decreases as $\frac{1}{N}$, but the computational cost grows linearly with N. Achieving higher accuracy (i.e., smaller variance) requires a large number of samples, which can be computationally expensive, especially when each sample is costly to compute.

Multilevel Monte Carlo Method

The key idea behind MLMC is to improve efficiency by performing simulations at multiple levels of resolution. Instead of simulating all samples at the highest level of accuracy (which is computationally expensive), MLMC introduces coarser levels where simulations are cheaper, and refines the result by focusing on the differences between successive levels. This reduces the overall computational cost while maintaining accuracy.

Let P_ℓ denote the approximation of the quantity P at level ℓ, where higher levels correspond to finer approximations (smaller timesteps, higher resolutions, etc.). The expected value of P can be rewritten using the telescoping sum:

$$\mathbb{E}[P] = \mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{\ell=1}^{L} \mathbb{E}[P_\ell - P_{\ell-1}],$$

where P_L is the approximation at the finest level, and L is the maximum level. Here, P_0 is a coarse approximation, and each term $P_\ell - P_{\ell-1}$ represents the correction between two successive levels.

The idea is to estimate each term $\mathbb{E}[P_\ell - P_{\ell-1}]$ independently using Monte Carlo sampling. The key advantage of MLMC is that the coarser levels (lower ℓ) are much cheaper to simulate, allowing us to use more samples at those levels, while the finer levels (higher ℓ) require fewer samples due to their low variance.

MLMC Estimator

Let N_ℓ be the number of samples used at level ℓ. The MLMC estimator for $\mathbb{E}[P]$ is given by:

$$\hat{P}_{\text{MLMC}} = \frac{1}{N_0} \sum_{i=1}^{N_0} P_0^{(i)} + \sum_{\ell=1}^{L} \left(\frac{1}{N_\ell} \sum_{i=1}^{N_\ell} \left(P_\ell^{(i)} - P_{\ell-1}^{(i)} \right) \right).$$

The first term corresponds to the coarse estimate at level 0, and the second term represents the sum of corrections across all levels. This approach significantly reduces variance while keeping the computational cost low.
In order to work properly, MLMC has to fit those conditions :

- The term $P_{l-1}^{(i)}$ has to be simulated in the same Monte Carlo simulation as $P_l^{(i)}$ so that they are highly correlated and the variance of the difference is low. It is generally done by taking a subset of PVs used by $P_l^{(i)}$. That way we add as less variance as possible with each level.
- $P_{l-1}^{(i)}$ must have the same expectation as the same term calculated in the previous level. That way the bias is the last level's bias which is the most refined level.

Those conditions are in contradiction and represent a technical challenge.

Complexity and Cost Analysis

The goal of MLMC is to balance the computational cost and accuracy by selecting appropriate numbers of samples N_ℓ at each level. The variance of the correction term $P_\ell - P_{\ell-1}$ decreases as ℓ increases, allowing fewer samples at higher levels.

The total computational cost $\mathcal{C}$ of MLMC is the sum of the costs across all levels:

$$\mathcal{C}_{\text{MLMC}} = \sum_{\ell=0}^{L} N_\ell \cdot c_\ell,$$

where c_ℓ is the computational cost per sample at level ℓ. Typically, the cost per sample increases with ℓ due to finer resolution, but the number of samples N_ℓ decreases at higher levels, leading to a significant reduction in total cost.

In contrast, the cost of a standard Monte Carlo simulation with the same level of accuracy would be much higher, scaling as $\mathcal{O}(\epsilon^{-3})$ for error tolerance ϵ. In MLMC, if the assumptions are respected, by optimizing N_ℓ across levels, the computational complexity can be reduced to $\mathcal{O}(\epsilon^{-2})$.

Our specific case

In our case, the parameters that vary across levels are the number of Monte Carlo paths and the number of time steps ([7]). The scaling of these parameters is determined by the user applying the method.
The maximum number of levels L is also chosen by the user.

5.2 Implementation of the interpolation in MLMC

As we aim to reduce computational costs, we need to choose what term MLMC will receive the interpolation treatment. For example we could use our interpolation in level 0, in a single level after level 0, in all levels... It depends how much the interpolation costs in time and how granular the result will be.

We need to watch how granular the interpolation is because there could be almost no space left for simulated points if the dates are saturated by interpolated points.

5.3 MSE estimation

We can read from [1] that if the number of level we use is L, then we can express the MSE as :

$$MSE = \mathbb{E}[(P_L - P)^2] = (\mathbb{E}[P_L] - \mathbb{E}[P])^2 + \sum_l \mathbb{V}[P_l - P_{l-1}] = \beta_L^2 + \sum_l \sigma_l^2$$

Using that and as assumed in [1] we can assume that MSE is only affected by the bias of the top level.

5.4 Bias estimation

We consider the following setup:

- N paths indexed by i,
- using a uniform schedule with density d dates per unit of time, where $d > 1$.

Let m be an integer such that $m > 1$ and divides d. We are interested in computing the bias in an XVA calculation using a multilevel Monte Carlo approach.

Let's start by evaluating the difference of an XVA (X) computed at different frequencies:

$$X(d) - X\left(\frac{d}{m}\right) = \frac{1}{N}\sum_{i=1}^{N} X_i(d) - \frac{1}{N}\sum_{i=1}^{N} X_i\left(\frac{d}{m}\right)$$

This can be expanded as:

$$X(d) - X\left(\frac{d}{m}\right) = \frac{1}{N}\sum_{i=1}^{N} (b_i(d) + X_i(\infty)) - \frac{1}{N}\sum_{i=1}^{N} \left(b_i\left(\frac{d}{m}\right) + X_i(\infty)\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left(b_i(d) - b_i\left(\frac{d}{m}\right)\right)$$

Now, assuming that the bias $b_i(d)$ scales as $\frac{\alpha_i}{d}$, we can write:

$$|X(d) - X\left(\frac{d}{m}\right)| = \left|\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\alpha_i}{d/m} - \frac{\alpha_i}{d}\right)\right| = \left|(m \pm 1)\,\frac{1}{N}\sum_{i=1}^{N} b_i(d)\right|$$

Thus, the Monte Carlo estimate of the bias is given by:

$$\frac{1}{N}\sum_{i=1}^{N} b_i(d) = \frac{|X(d) - X\left(\frac{d}{m}\right)|}{m \pm 1}$$

The remaining task is to determine the sign in $m \pm 1$. Since the sign of $b_i(d)$ is unknown and could flip, we assume that the sign flips half the time and we take the expectation. This leads to the following expression for the expected bias $\beta(d)$:

$$\beta(d) \approx \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} b_i(d)\right] = \frac{1}{2}\cdot\frac{|X(d) - X\left(\frac{d}{m}\right)|}{m+1} + \frac{1}{2}\cdot\frac{|X(d) - X\left(\frac{d}{m}\right)|}{m-1}$$

Simplifying this expression, we obtain:

$$\beta(d) = \frac{m}{m^2 - 1}\left|X(d) - X\left(\frac{d}{m}\right)\right|$$

In practice, we will perform a single simulation and then downsample the dates of the paths to obtain the downsampled values.
While this approach could be criticized due to the high correlation between paths, it is not an issue for our purposes. Since our goal is to estimate the integration bias, eliminating Monte Carlo noise bias is not a concern in this case.

5.5 Standard deviation estimation

The standard deviation estimation

6 Results

6.1 Results methodology

We need to see if our interpolation leads to stable results ie we get similar results if we use different random numbers. We also want to see the time cost and memory consumption with the same tests.

In order to test that the methodology is the following :

- Choose $N > 100$

- Repeat N times:
  - Generate a CVA value
  - Store CVA value, used memory peak and calculation time
  - Change random numbers seed

We do this with and without interpolation and we visualize the distribution of the obtained CVAs as well as time and memory. In order to compare comparable things we choose our interpolation parameters so that the total number of points is the same. This can be viewed as replacing simulated points with interpolated points thus replacing the simulation cost with the interpolation cost in time and memory. We expect the memory to be similar since the result will be stored in similar size objects. We use the peak memory because this is what could make the machines crash when running such calculations.

The time is more interesting because it will tell us whether the interpolation cost in time is worth it.

We chose $N = 200$ for those results.

6.2 Stability results

This plot describes the distribution of the obtained CVA numbers otained by the previously described method.

Figure 7: CVA values histogram for different simulations

We see that the distributions match which means we achieve a similar RMSE.

6.3 Memory results

This is similar to the previous plot but represents the peak memory distribution during each calculation.

Figure 8: Peak memory values histogram for different simulations

As we expected the peak memory used match for similar size results.

6.4 Time results

This is the histogram plot of the times taken by each calculation to run.

Figure 9: Calculation time values histogram for different simulations

We can see that the times taken are around 3mins and the interpolation seems to be faster in average than the simulation.

We need to add explanation elements here. The averages are close to each other and the results would be more expressive if they were far away but we have to note that 3mins is almost nothing compared to the real tasks that run in production and this was chosen on purpose to be able to run the 200 calculations in a finite time for this experiment.

Since the standard deviation of the time is quite low I ran some calculations with larger tasks and found out that simulated points are at least 1.5 times faster to generate than simulated points. I don't prove how I obtained this result here since it really depends on the engine and the used models. Caution : this number doesn't mean that the whole calculation will be 50% faster. The estimated time reduction varies for counterparties so we cannot make a general assumption on its value except it always faster.

References

[1] Internal document.

[2] Leif B.G. Andersen, Michael Pykhtin, and Alexander Sokol. Rethinking margin period of risk. *SSRN Electronic Journal*, 2016. Available at SSRN: https://ssrn.com/abstract=2719964.

[3] Christoph Burgard and Mats Kjaer. Derivatives funding, netting and accounting. *SSRN Electronic Journal*, March 20 2017. Available at SSRN: `https://ssrn.com/abstract=2534011`.

[4] Christian Böinghoff and Martin Sprenger. Alternatives to log-normal and normal models in market risk: The displaced historical simulation and the mixed model. *SSRN Electronic Journal*, 2020. Available at SSRN: `https://ssrn.com/abstract=3681809`.

[5] M. B. Giles. Multilevel monte carlo methods. 2018.

[6] Andrew Green. *XVA: Credit, Funding and Capital Valuation Adjustments*. Wiley, 2015.

[7] Markus Hofer and Patrik Karlson. Accelerating xva calibration using multilevel monte carlo. *Wilmott*, 2018. First published: 19 July 2018, DOI: `https://doi.org/10.1002/wilm.10689`.

[8] Gilles Pagès. *101 quizz qui banquent: Mathématiques et finance sont-elles indépendantes ?* Poche – Grand livre, 2012. Published: 26 October 2012.

[9] Steven E. Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer, New York, 2004.

[10] S. S. Szyszkowicz and H. Yanikomeroglu. Limit theorem on the sum of identically distributed equally and positively correlated joint lognormals. *IEEE Transactions on Communications*, 57(12):3538–3542, 2009. Available at: `https://www.sce.carleton.ca/~sz/LimitTheoremSLN.pdf`.

[11] Steven Zhu and Michael Pykhtin. A guide to modeling counterparty credit risk. *GARP Risk Review, July/August 2007*, 2007. Available at SSRN: `https://ssrn.com/abstract=1032522`.