POLITECNICO
MILANO 1863

Financial Engineering 2024/25

Final Project Group 6B - Report

When rates are squared

Donato Augenti - 10793800
Dario Delmonaco - 10679120
Ulysse Grimault - 11056380
Davide Rota - 10796557

Abstract

This project focuses on pricing a structured certificate with a Constant Maturity Swap (CMS) as the underlying. Instead of using standard Monte Carlo or tree-based methods, we considered a bounding approach (upper and lower bounds) to frame the price based on CMS option values options.

A key part of the pricing process is building the implied volatility curve, which we derive from market-quoted Euribor swaption data using two models: the SABR model and the Additive Bachelier model.

For each model, we explore two calibration strategies: a direct method that estimates all parameters at once, and a cascade method that uses known relationships between parameters to simplify the process.

Contents

Disclaimer: *all computations have been done with an unitary notional.*

The SABR model

The SABR (Stochastic Alpha Beta Rho) model is a stochastic volatility model which takes into account the following dynamics for underlying forward rate and its volatility:

$$\begin{cases} dF_t = \alpha_t F_t^{\beta}\, dW_1(t) \\ d\alpha_t = \nu \alpha_t\, dW_2(t) \\ dW_1(t) \cdot dW_2(t) = \rho\, dt \\ \alpha_0 = \alpha \end{cases}$$

where F_t is the forward swap rate at time t for a given start date and tenor, while $W_1(t)$ and $W_2(t)$ are two different and correlated Brownian motions.

The main property of this model we want to introduce to calibrate is the fact that it has three parameters:

- α: is a process (time varying) that represents the volatilty of the forward rate.
- β: is a constant paramter that helps us to look at the dependency between the volatility α and the process F_t.
- ρ: is the correlation coefficient between the two driving Brownian motions defined above.

Moreover, ν is another element and it is the volatility of the volatility (*volvol*).

β is frequently pre-selected from *a priori* considerations because there is a large degree of redundancy between β and ρ (both affect the volatility smile in similar ways). *In our analysis, β will be equal to zero representing a stochastic normal model.* For these reasons, the calibration of the SABR model is performed on α, ρ and ν.

The SABR model accurately fits the implied volatility curves observed in the market for any single exercise date and allows to predict the correct dynamics of the implied volatility curves.

The forward (start) swap rates

To calculate forward swap rates, we first need to provide a bootstrap of the discount factors (and eventually of the rates), where the starting date is the 30th April 2021. To construct it, we need to take into account the values inside the Excel file *EURswptVols*, in the page *Curve*: from this, we consider the first table, in which there is the Euribor 6m forwarding curve (with deposit, forward rate agreements and swap).

The very first discount factor on the 4th of November 2021 was computing using the deposit rate quoted in the market :

$$B(t_0, t_1) = \frac{1}{1 + \delta_1 L(t_0, t_1)}$$

To compute the next discount factors for FRA instruments we used the quoted forward rates quoted in the market $L(t_0, t_i, t_j)$:

$$L(t_0, t_i, t_j) = \frac{1}{\delta(t_i, t_j)} \cdot (\frac{B(t_0, t_i)}{B(t_0, t_j)} - 1)$$

$$\text{so} : B(t_0, t_j) = \frac{B(t_0, t_i)}{1 + \delta(t_i, t_j) L(t_0, t_i, t_j)}$$

where $B(t_0, t_i)$ denotes the discount factor at the settlement date of the FRA. It was obtained by interpolation (or extrapolation, if needed) on the zero rate curve constructed up to that point in the bootstrap procedure.

To boostrap the discount factors after the 4th of May 2023 we used the market-quoted SWAP rates. Note that we did not consider the quoted swap rate of the 4th of May 2023 to avoid repetition with the quoted FRA rate on this very same date, we choose to remove this one because it is less liquid than FRA.

$$S(t_0, t_0, t_N) = \frac{1 - B(t_0, t_N)}{\sum_{i=1}^{N} \delta(t_{i-1}, t_i) B(t_0, t_i)}$$

$$\text{so} : B(t_0, t_N) = \frac{1 - S(t_0, t_0, t_N) \cdot \sum_{i=1}^{N-1} \delta(t_{i-1}, t_i) B(t_0, t_i)}{1 + \delta(t_{N-1}, t_N) S(t_0, t_0, t_N)}$$

where the $B(t_0, t_i)$ were obtained by linear interpolation on the zero rate curve.

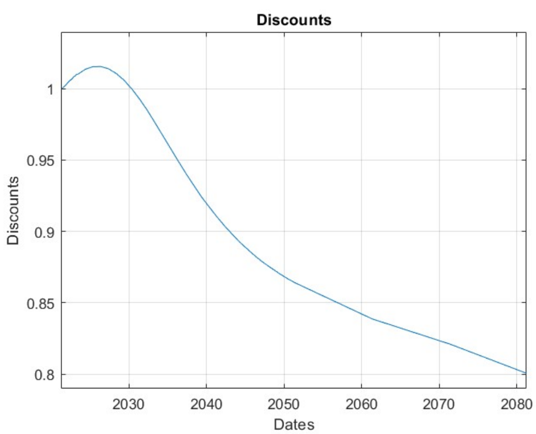


Figure 1: Bootstrap curve for the discount factors

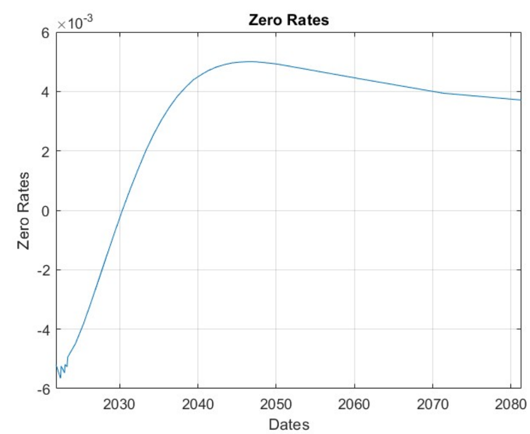


Figure 2: Bootstrap curve for the zero rates

It can be observed that up to the 9th year the rates in the table have negative values: this is reflected in having discount factors bigger than 1, which is also confirmed in the shape of the curve; it starts having a discount factor equal to 1 at inception, than they are higher than the unitary value up to the ninth year and after that date they are lower then 1. 9 year is a crucial date, since at that date the value of the rate is nearly 0, so the correspondent discount factor is really close to 1.

Once having computed the bootstrap, it is possible to calculate the forward swap rates: using the Matlab function `get_fwd_swap_rate` we started from the data of the bootstrap and after having computed the correspondent BPV, we obtained the forward swap rate for a given expiry and tenor.

In the following table are shown some forward swap rates for relevant maturities and tenors:

	Tenor 1y	**Tenor 5y**	**Tenor 10y**
1m	-0.6029%	-0.3061%	0.0767%
1y	-0.4393%	-0.1663%	0.2041%
5y	0.1581%	0.4509%	0.6803%
10y	0.7994%	0.8981%	0.8724%

Table : Forward (start) swap rates for significative expiries-tenors

Direct calibration

The aim of this section is to calibrate the three parameters introduced above (α, ν, ρ) for different couples maturity-tenor, considering a subset of the ones we have in the table *Vol* of the market (swaption) at-the-money volatilities.
The calibration, for a given couple maturity-tenor, is performed with the minimization of an objective function, which is the sum of the squared differences between the implied normal volatilities and the quoted market ones for the given relative strikes.
In the Excel file, there are two grids: one contains the at the money swaption normal volatilities (in bps) for different expiries and tenors, while the second one shows the offsets (in bps) of the volatilities when the relative strike changes. To obtain the effective values of the volatilities, for a given couple expiry-tenor and for a certain relative strike, it is necessary to sum the offset to the correspondent at the money volatility in the other table.
The strikes can be found from the tables in the Excel file, while for the implied volatilities we considered the following formulas:

$$\sigma(K) = \alpha \cdot \frac{\xi}{\hat{x}(\xi)} \cdot \left(1 + \frac{2 - 3\rho^2}{24} \cdot t \cdot \nu^2\right) \tag{1}$$

$$\text{s.t.} \quad \sigma_{ATM} = \alpha \cdot \left(1 + \frac{2 - 3\rho^2}{24} \cdot t \cdot \nu^2\right) \tag{2}$$

with:

$$\xi = \frac{\nu}{\alpha} \cdot (f - K) \quad , \quad \hat{x}(\xi) = \log\left(\frac{\sqrt{1 - 2\rho\xi + \xi^2} + \xi - \rho}{1 - \rho}\right)$$

where $\sigma(K)$ is the implied normal volatility of the model for a certain strike K, while σ_{ATM} is the at-the-money volatility, f is the swap forward rate value and t exercise time to maturity year-fraction.

In order to compute all the values we need, we considered the forward swap rate f for a given expiry and tenor, then we calculated the absolute strike with the following relation: $K = f + k$, where k is the relative strike (in percentage) and K is the absolute strike.
Thanks to the function `sigma_computation`, which computes the $\sigma(K)$ of formula (1), we have defined the

objective function described above. Then, `fmincon` was used to minimize the objective function, to obtain the optimal calibration parameters.

Some constraints have been imposed in order to have an admissible solution:

$$\begin{cases} \alpha \geq 0 \\ \nu \geq 0 \\ \rho \in [-1, 1] \end{cases}$$

are constraints for the admissibility of parameters, since the volatility and the volatility of the volatility must be positive (or at most equal to 0), while the correlation coefficient has a value inside the closed interval from -1 to 1.

NOTE 1: In our Matlab code, when constructing the function `fmincon`, we did not inserted any of these three constraints. We tried to perform the calibration also with these upper and lower bounds for the parameters, but the results were far worse than if trying without these constraints. However, at the end of the computations, we checked that all the values obtained could satisfy these conditions and any value for the parameters was in the existence domain for each one of them.

Also the following non-linear inequalities are needed as constraints, since these define the values for which the *square root* and the *log* are well defined:

$$\begin{cases} 1 - 2\rho\xi + \xi^2 \geq 0, \\ \dfrac{\sqrt{1 - 2\rho\xi + \xi^2} + \xi - \rho}{1 - \rho} > 0. \end{cases}$$

These inequalities may not be essential in the calibration, but we prefer to also add this to have a feasible solution.

The last constraint we added is formula (2), a nonlinear equation for the σ_{ATM}: $\sigma_{ATM} = \alpha \cdot \left(1 + \frac{2-3\rho^2}{24} \cdot t \cdot \nu^2\right)$. This method yields at-the-money volatilities that align precisely with observed market quotations. It is commonly employed in the pricing of swaptions, where at-the-money volatilities are the most frequently quoted and are important to match.

NOTE 2: The values for the initial guess of `fmincon` are not completely random but we chose them after an analysis of the objective function: looking at its surface, it is possible to see there are lots of local minima, so the choice of the initial guess changed the solution dramatically; it was necessary to find a reasonable value in order to arrive to the global minimum of the function. Furthermore, another approach we followed to find a possible initial guess was to randomize it and check what result was found for the objective function, in correspondence of which one the lowest value was reached.

In the following tables are shown some representative values, given a copule expiry-tenor, for the three parameters, obtained by the calibration described above:

α	**1y**	**5y**	**10y**
1m	0.0012	0.0025	0.0038
1y	0.0018	0.0037	0.0048
5y	0.0051	0.0054	0.0055
10y	0.0055	0.0055	0.0055

Table : Parameter alpha

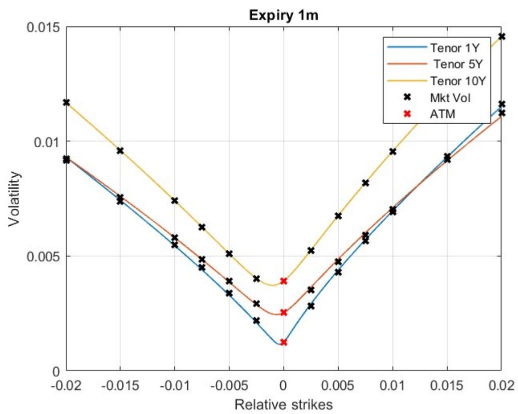
ν	**1y**	**5y**	**10y**
1m	2.2071	1.6550	1.9590
1y	1.0902	0.8639	0.7679
5y	0.4040	0.3010	0.2759
10y	0.2041	0.2073	0.1908

Table : Parameter nu

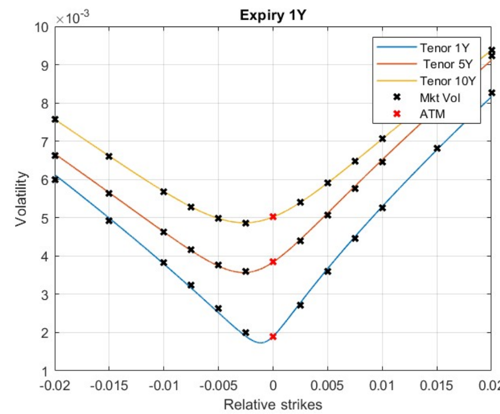
ρ	**1y**	**5y**	**10y**
1m	0.4448	0.3028	0.3583
1y	0.4713	0.4235	0.2802
5y	0.5661	0.3508	0.1762
10y	0.5189	0.3273	0.1581

Table : Parameter rho

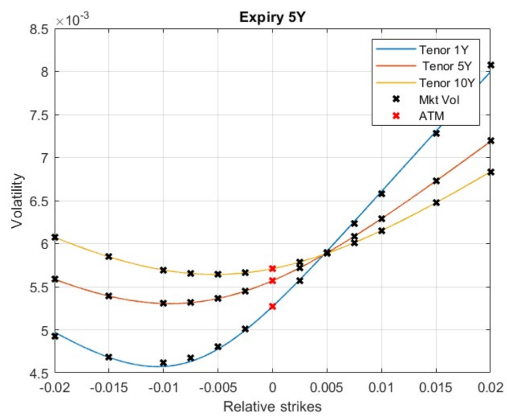
Once completed the calibration of the three parameters of the SABR, the implied volatilities can be found thanks to the analytical formula.
We are going to plot the implied volatility smiles, showing the results of some couples of relevant expiry-tenor. On a given plane, it is possible to see the changes of the smile, given a fixed expiry (or tenor), making a comparison among them and the market volatility smile.



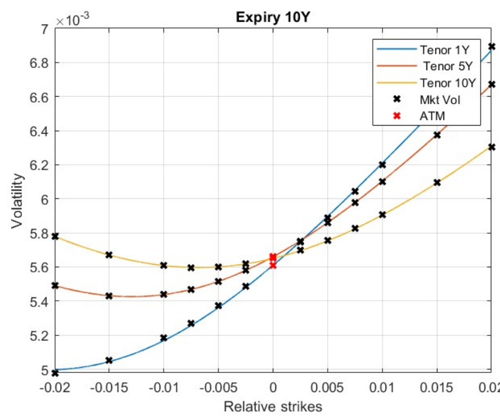
(a) Expiry 1m



(b) Expiry 1Y

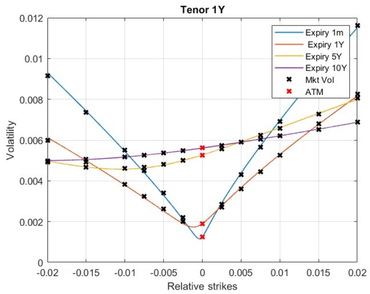


(c) Expiry 5Y

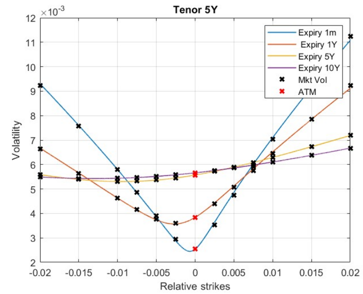


(d) Expiry 10Y

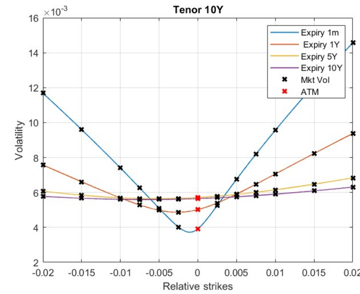
Figure 3: Volatility smiles fixing expiry



(a) Tenor 1Y



(b) Tenor 5Y



(c) Tenor 10Y

Figure 4: Volatility smiles fixing tenor

Observations: In general SABR implied volatility smiles match almost excatly the market European swaption implied volatilities. By fixing a short expiry, we observe that increasing the tenor results in an upward shift of the volatility smile while preserving its overall shape. Conversely, when fixing a longer expiry, shorter tenors exhibit lower implied volatilities for deep in-the-money swaptions (i.e., negative relative strikes) and higher volatilities for deep out-of-the-money swaptions compared to longer tenors. Furthermore, when the tenor is held constant, the volatility curve tends to display the typical "smile" pattern for short expiries regardless of the tenor, whereas this curve progressively flattens as the expiry increases.

Without linear constraint σ_{ATM}

We also tried to calibrate the three parameters without including the formula (2), the linear constraint for the at-the-money volatility, too see how the results could change.
In this case, the at-the-money implied volatility of the model does not match market quotations as well as the previous case.

Here follows the plot for these new results:

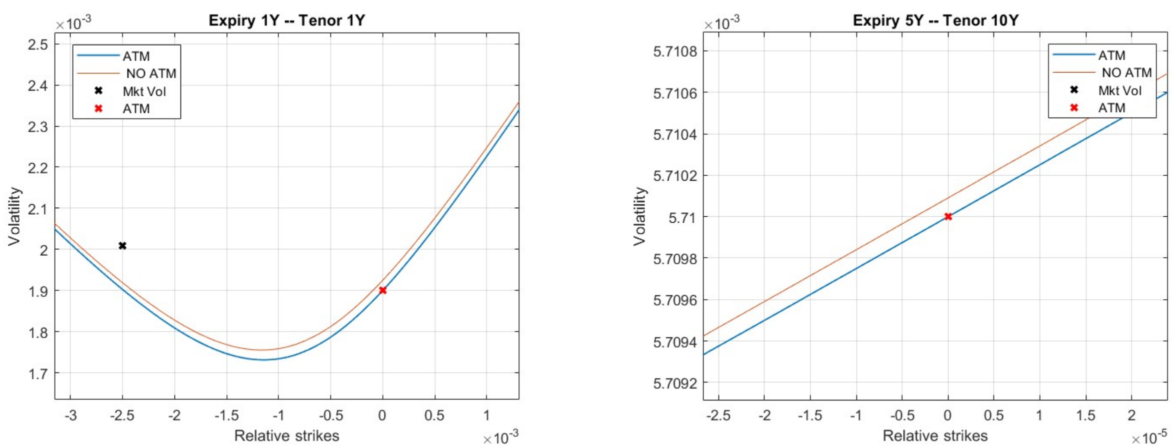


Figure 5: Volatility smiles imposing the ATM-vol. constraint (ATM line) and without imposing it (NO ATM)

Calibrate first ν' and ρ on the normalized volatilities and then α and ν

Here, the aim is still to calibrate the same parameters as before, but in two different steps.
In the first step, a new parameter is introduced: $\nu' = \frac{\nu}{\alpha}$ and the minimization of the objective function is made to find a value for this parameter and ρ; to do this, we rely on normalized volatilities: $\frac{\sigma(K)}{\sigma_{ATM}}$.
Once computed, α and ν can be exploited substituting the ν' relation into formulas (1) and (2).

We obtained the normalized sigma for every couple maturity-tenor by dividing the market volatility for the corresponding atm-sigma. After having computed the corresponding forward swap rate and the absolute strike as we did in the previous case, we construct the objective function: it is the same as above, with the only difference given by the function `sigma_computation_2` which compute the value $\frac{\sigma(K)}{\sigma_{ATM}} = \frac{\xi}{\hat{x}(\xi)}$ where now $\xi = \nu' \cdot (f - K)$.

Applying the function `fmincon` to this objective function and the same constraints we have considered in direct calibration, it is possible to find the values for the parameters ν' and ρ, which are saved since they would be used in the next step. Then, we solved a system of two equations:

$$\begin{cases} \nu' = \dfrac{\nu}{\alpha} \\ \sigma_{ATM} = \alpha \cdot \left(1 + \dfrac{2 - 3\rho^2}{24} \cdot t \cdot \nu^2\right) \end{cases}$$

for the variables α and ν, also relying on the `fsolve` Matlab function.

An advantage of this method is that the atm constraint is implied in the calibration, since it is not imposed but must be satisfied to get the three parameters.
Moreover, this method is computationally more efficient since now there is not a single optimization of three parameters at the same time, but a cascade: before two are obtained and then, simply solving a system of equations the results are determined.

In the following table are shown the values of the parameters obtained via cascade calibration, for some relevant couples expiry-tenor (the same as for the previous table):

α	**1y**	**5y**	**10y**
1m	0.0012	0.0025	0.0038
1y	0.0018	0.0037	0.0048
5y	0.0051	0.0054	0.0055
10y	0.0055	0.0055	0.0055

Table : Parameter alpha

ν	**1y**	**5y**	**10y**
1m	2.2071	1.6550	1.9590
1y	1.0902	0.8639	0.7679
5y	0.4040	0.3010	0.2759
10y	0.2041	0.2073	0.1908

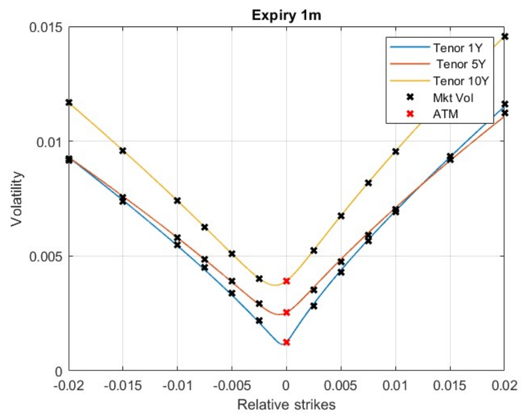
Table : Parameter nu

ρ	**1y**	**5y**	**10y**
1m	0.4448	0.3028	0.3583
1y	0.4713	0.4235	0.2802
5y	0.5661	0.3508	0.1762
10y	0.5189	0.3273	0.1581

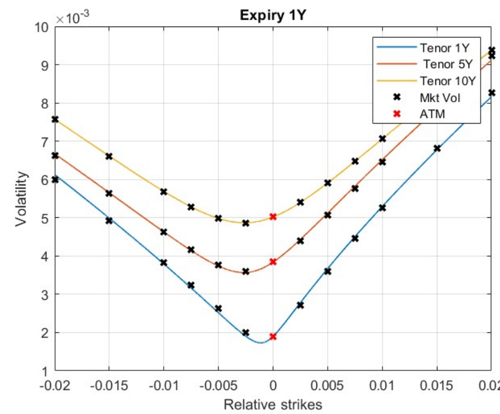
Table : Parameter rho

Plots of the implied volatility curves and comparison

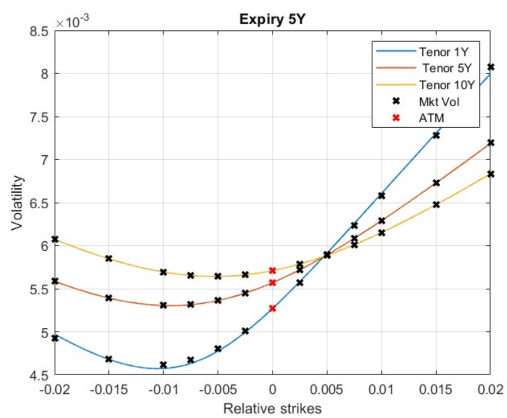
As for direct calibration, it is possible to plot volatility smiles for some relevant expiries or tenors. In the following plots are compared market smiles with respect to the ones resulting from the calibrated model.



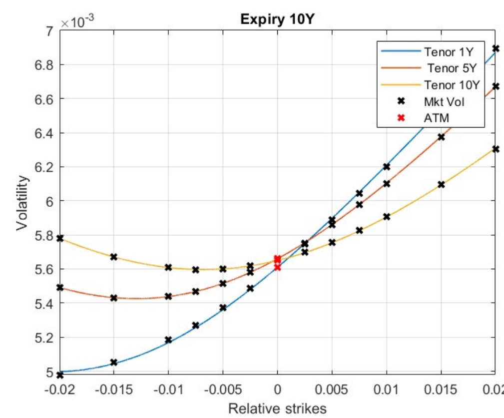
(a) Expiry 1m



(b) Expiry 1Y

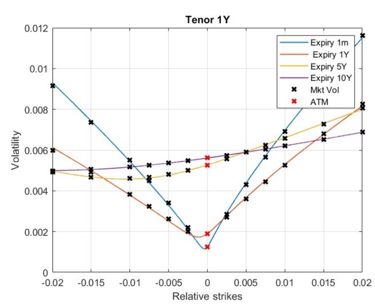


(c) Expiry 5Y

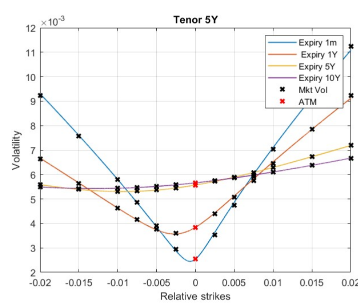


(d) Expiry 10Y

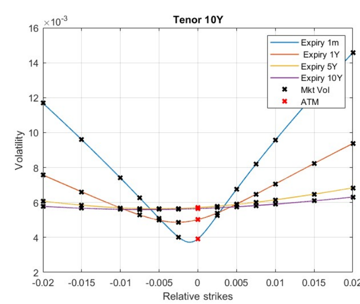
Figure 6: Smiles fixing expiry



(a) Tenor 1Y



(b) Tenor 5Y



(c) Tenor 10Y

Figure 7: Smiles fixing tenor

Observations: We can see that the same observations exposed for the Direct calibration approach still hold.

The Additive Bachelier

The additive Bachelier approach considers a dynamics for the forward swap rate equal to:

$$F_t = F_0 + I \cdot W_t$$

where I is the implied volatility and W_t is a Brownian motion.

Under this dynamics, the price of an European swaption, normalized by B_0, the square root of the maturity and the forward BPV is given from this equation:

$$\frac{1}{B_0 \cdot BPV_0 \sqrt{t}} \cdot B_0 \cdot BPV_0 \, \mathbb{E}_0\left[(F_t - K)^+\right] = c_b(\frac{x}{\sqrt{t}}; I)$$

$$\text{with} \quad c_b(y; I) := -y\, N\left(-\frac{y}{I}\right) + I \cdot \frac{1}{\sqrt{2\pi}}\, e^{-\frac{y^2}{2I^2}}$$

where x is the moneyness, N is the CDF of a stardard gaussian random variable and y, the moneyness degree, is defined as $\frac{x}{\sqrt{t}}$. c_b can be considered as the normalized market price in our calibration, while the model price can be obtained from this formula:

$$c(x; t; \mathbf{p}) = \mathbb{E}\left[c_b\left(\frac{x}{\sqrt{t}} + \sigma(G-1);\, \sigma\sqrt{G}\right)\right] \tag{3}$$

where G is a positive random variable with mean equal to 1, variance κ and Laplace exponent *l(u; k; α)*, while $\mathbf{p}$ is a set of three parameters: the volatility σ, η and κ.

This price is equivalent to the one obtained via Lewis formula (*), which is:

$$c(x; t; p) := \mathbb{E}[(z-y)^+] = \frac{e^{-ya}}{2\pi} \int_{-\infty}^{+\infty} \frac{e^{-i\xi y}}{(i\xi + a)^2}\, \phi(\xi - ia)\, d\xi \tag{4}$$

where a is equal to $\frac{1}{2}$, ϕ is the characteristic function of z and z is equal in law to $\frac{F_t}{\sqrt{t}}$.

When model price is computed, we use the Lewis formula for computational efficiency.

(*) Verification of consistency of the two prices for a given set of parameters

To check numerically the consistency between the formulas (3) and (4), we implemented the Matlab function `Real_model_price`, which determines the expected value of $(c_b\left(\frac{x}{\sqrt{t}} + \sigma(G-1);\, \sigma\sqrt{G}\right))$ as the following integral, computed in Matlab by `quadgk` :

$$\int_0^{+\infty} c_b\left(\frac{x}{\sqrt{t}} + \sigma(g-1);\, \sigma\sqrt{g}\right) \cdot f(g) dg$$

where $f(g) := \dfrac{e^{-\frac{(-1+g)^2}{2kg}}}{\sqrt{2\pi}} \sqrt{kg^3}$ is the probability density function of the random variable G; moreover, we implemented the function `Model_price` which computes the Lewis formula described by (4) (a more detailed description of this function is in the following section).

We then computed the prices for the parameters $\sigma = 0.1,\ \eta = 1,\ \kappa = 1$ and as time-to-maturity of the swaption $ttm = 1$ for the strikes quoted on the market; the error we found was of the order of 10^{-10} for each strike, so we can conclude that the two formulas are equivalent.

Direct calibration

The aim of this section is to calibrate the three parameters (σ, η, κ) for the same couples maturity-tenor we considered for the previous model. The calibration, for a given couple maturity-tenor, is performed with the minimization of an objective function, which is the sum, for the given strikes, of the squared differences between the model prices and market prices of European swaptions, both normalized by discount factor, forward basis point value and the square root of the time-to-maturity.

After having computed the forward swap rate and the moneyness, we calculate the two different prices we need to plug in the objective function thanks to two different Matlab functions:

- `Model_price` can compute the normalized swaption price from Lewis formula, solving the integral by FFT approach or numerical integration (using `quadgk` function of Matlab); to be the more general as possible, we set an input of this function as a flag, for which by only changing this we can have the result using one technique or the other.
- `Bachelier_swaption_price` compute the normalized price by Bachelier formula taking in input the moneyness degree and market implied volatility for the desired maturity-tenor of the swaption.

The following contraints must be considered if we want to get an admissible solution:

$$\begin{cases} \sigma > 0 \\ k > 0 \\ \eta \in \mathbb{R} \end{cases}$$

Differently from the previous model, here we don't need to adjust anything to have the optimal solution (as the *Note 1* above). These boundaries are implemented in the *lower bound* input of `fmincon`.
`fmincon` is the Matlab function used to minimize the objective function, taking into account all the constraints we need : the aim is to directly find the optimal values for all the three parameters for a given maturity-tenor at the same time.
The next step is the calculation of the implied volatility with `fsolve`: to compute it, it is necessary to equalize the model price and the market price; the implied volatility is the second parameter we have to plug in inside the Bachelier formula c_b, then solving the equation with respect to this incognita it is possible to find the implied volatility.

The choice of the initial guess was not trivial: we started considering a random initial guess for the first couple expiry-tenor (the one in the first row and first column in *GridA* of *EURswptVols*), then take the resulting parameters of this iteration and plugging them as initial value for all the other couples where the expiry or the tenor was the same; for all the other couples, we relied on the results obtained from the first column (same expiry but the lowest tenor possible) and plugged that result as initial guess: unfortunately, the result was bad and so we needed to change our approach.
We searched for an initialization smile by smile (one for couple expiry-tenor), by randomizing the starting

guesses and saving the best ones: the choice was done considering the norm of the error and we took the one with the lowest. Moreover, we noticed that the model is not robust with respect to the initial guess: by slightly varying it, the norm of the error can change much.

In the following table are shown the results for some significative values of the three parameters calibrated, for some relevant couples expiry-maturity (we consider a close in time date, a middle distance date and a far date from the settlement date).
Disclaimer: the values of σ are expressed in bpv.

σ	**1y**	**5y**	**10y**
1m	0.06117	0.1842	36.475
1y	28.214	41.688	56.836
5y	55.315	62.724	64.785
10y	61.484	65.399	64.458

Table : Parameter sigma

κ	**1y**	**5y**	**10y**
1m	41.718	332.10	0.0524
1y	3.4137	2.6544	1.8027
5y	2.6795	2.3655	2.2074
10y	1.9802	2.7795	2.1880

Table : Parameter kappa

η	**1y**	**5y**	**10y**
1m	-21.5545	-1.6839	2.8104
1y	-0.2230	-0.3193	-0.1542
5y	-0.3730	-0.1658	-0.0999
10y	-0.2376	-0.1394	-0.0766

Table : Parameter eta

Calibrate first η and κ on the normalized volatilities and then σ on the ATM volatilities

Now, the aim is still to calibrate the same parameters as before, but in two different steps: first are calibrated two parameters η and κ and then the remaining one, σ.
To do this, we rely on normalized volatilities: $\frac{\sigma(K)}{\sigma_{ATM}}$.

First of all, we define the moneyness x as the value of relative strike K of the considered swaption and introduce the variable $\hat{y}$, which is defined as $\hat{y} := \frac{y}{\sigma_{\text{ATM}}}$; moreover, the volatility σ can be written as: $\sigma_{\text{ATM}} = \sigma \cdot I_0$, so the next step is to define I_0 as $I_0 = \sqrt{2\pi} \cdot \mathbb{E}\big[c_b(\eta(G-1), \sqrt{G})\big]$. In Matlab it has been implemented as $\sqrt{2\pi} \cdot$ `Model_price`, where in this case `Model_price` takes in input a σ equal to 1.

In the first step, what we need to do is to construct a new objective function that will be minimized giving us the first to parameters in this cascade resolution, η and κ.
This function is the sum of the squares of the differences between the market price and the model price, of European swaptions, both normalized by discount factor, forward basis point value and the square root of the time-to-maturity. As above, these two quantities can be computed with the following Matlab functions: `Model_price` and `Bachelier_swaption_price`; however, there are different input elements if compared to direct calibration: `Bachelier_swaption_price` is the same as before, only normalized by the σ_{ATM}, while `Model_price` takes in input as moneyness the product between $\hat{y}$ and I_0.
Through `fmincon`, it is possible to find the optimal values for unknown parameters η and κ.
These are later plugged in I_0, in order to solve also the second part of the cascade, in which we find the third and last paramter of the model, σ. σ is computed as the σ_{ATM} divided by the I_0 just re-defined after the first step.

The last part is the computation of the implied volatility: this has been considered as the σ_{ATM} multiplied by the normalized implied volatility, which we define with *H($\hat{y}$)*; the formula follows:

$$I_t(x) = \sigma_{\text{ATM}} \cdot H(\hat{y}) = \sigma_{\text{ATM}} \cdot H\left(\frac{x}{\sigma_{\text{ATM}}\sqrt{t}}\right)$$

To determine the normalized implied volatility, we imposed the equality between the model and the market prices as shown in the following formula:

$$\frac{1}{I_0}\,\mathbb{E}\left[c_b\left(\hat{y}\cdot I_0+\eta(G-1),\sqrt{G}\right)\right]=c_b\left(\hat{y},H(\hat{y})\right)$$

where the model price is determined using the calibrated parameter for a fixed maturity-tenor.

Here, $H(\hat{y})$ can be found by solving the equation with respect to this incognita: in Matlab it is possible to do it thanks to the function `fsolve`; then, once retrived this value, it is plugged in the formula of implied volatility to obtain it, one for each strike and couple expiry-tenor.

In the following table are shown the results for some significative values of the three parameters calibrated, for some relevant couples expiry-maturity.
Disclaimer: the values of σ are expressed in bp.

σ	**1y**	**5y**	**10y**
1m	13.453	15.098	39.522
1y	21.173	41.146	54.580
5y	54.497	57.471	60.438
10y	53.596	58.433	59.686

Table : Parameter sigma

κ	**1y**	**5y**	**10y**
1m	3.0186	0.1178	0.5986
1y	2.0385	1.7495	1.3799
5y	1.5636	0.8788	0.7087
10y	0.7518	0.7995	0.6554

Table : Parameter kappa

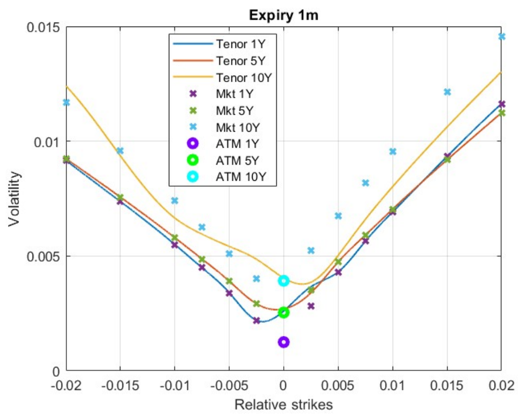
η	**1y**	**5y**	**10y**
1m	0.3557	-4.0813	-0.4510
1y	-0.2412	-0.3231	-0.2412
5y	-0.4387	-0.3741	-0.1990
10y	-0.6341	-0.3634	-0.1855

Table : Parameter eta

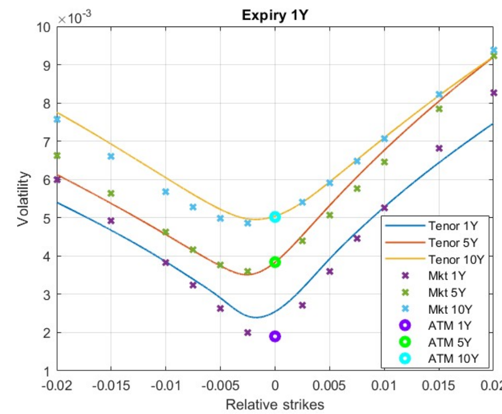
The cascade calibration should be more convenient than the direct one: a first big advantage is the computational time and complexity, the cascade should be faster since only two parameters are calibrated and only in a second time moment the third and last one, by simply solving an equation. Moreover, it is matched with the at-the-money values since the implied volatilities are calculated starting from the ATM ones.

Plots of the implied volatility curves and comparison

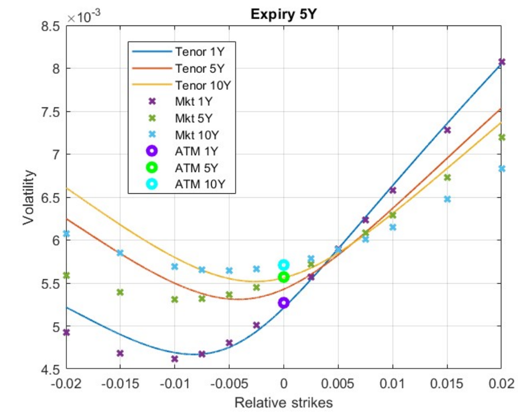
The following are the plots of the implied volatility smiles computed with additive Bachelier model, having fixed the expiry or the tenor in the direct approach.



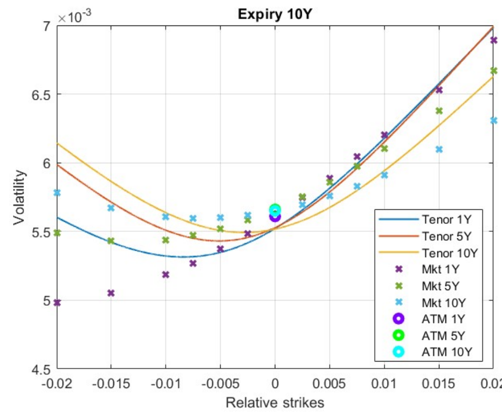
(a) Expiry 1m



(b) Expiry 1Y

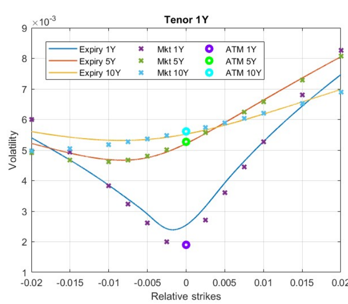


(c) Expiry 5Y

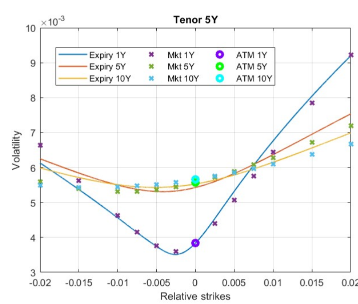


(d) Expiry 10Y

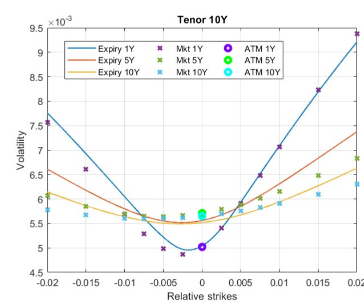
Figure 8: Smiles fixing expiry



(a) Tenor 1Y



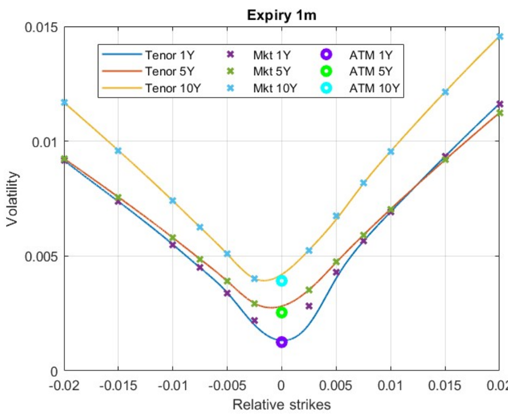
(b) Tenor 5Y



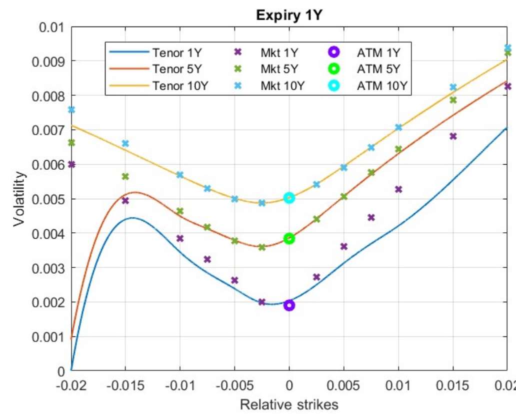
(c) Tenor 10Y

Figure 9: Smiles fixing tenor

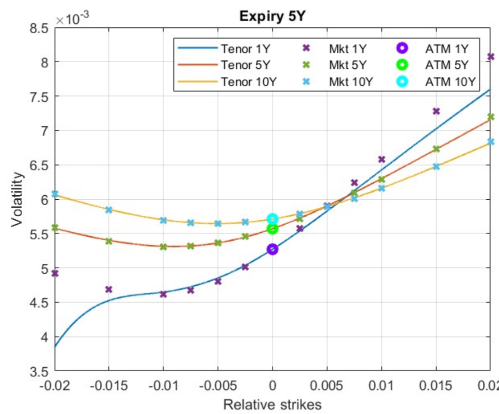
The following are the results with the cascade calibration:



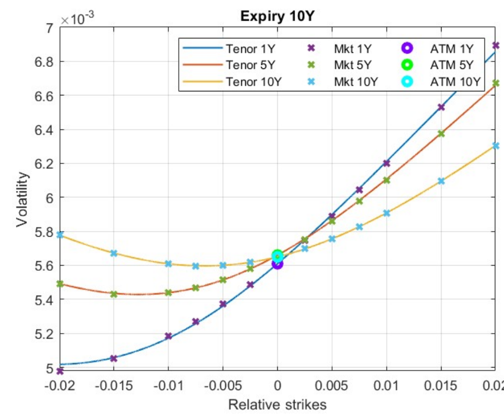
(a) Expiry 1m



(b) Expiry 1Y

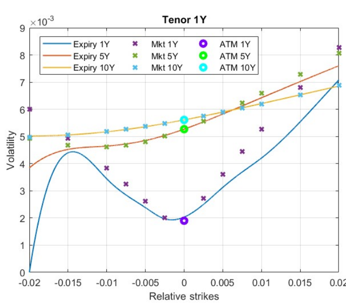


(c) Expiry 5Y

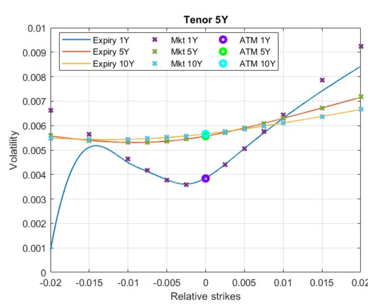


(d) Expiry 10Y

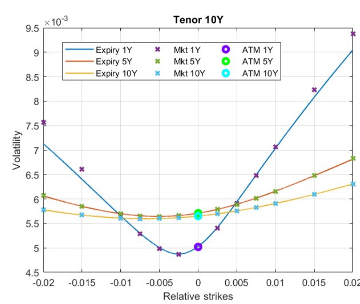
Figure 10: Smiles fixing expiry



(a) Tenor 1Y



(b) Tenor 5Y



(c) Tenor 10Y

Figure 11: Smiles fixing tenor

Observations: The volatility smiles obtained using the cascade approach are much closer to the observed market data than the direct approach. However, some inaccuracies remain, particularly for the lower relative strikes.11.

In a more general comparison, the smiles obtained with the SABR model are much closer to the market

volatility values if compared to the Additive Bachelier ones; this means that the parameters calibrated with the SABR are able to explain better the behaviour of the market and reproduce it in a more reasonable way than the Additive Bachelier model. The SABR model is so good especially because the calibration is directly made on the implied volatilities and not passing through the prices, as happens in Bachelier.
This result can also be emphasized by a computational component because it may be possible that the optimizations made in the code for SABR model are much more effective than the Bachelier part, so we can obtain parameters that fit in a more reliable way the market, also for the at the money values.
In particular, the cascade calibration gives an improvement with respect to the direct calibration of the three parameters of Additive Bachelier, even if for low relative strikes some issues are still present. The smiles of the Bachelier model are obtained by minimizing an objective function which takes into account not the difference of the volatilities, but of the prices: it is necessary to extract the implied volatility, which is likely less accurate than the other model.

Furthermore, the fit of the model results are quite similar if the expiry or the tenor changes: there is no significant difference when we fix one of these two dates and see the behaviour of the smiles with this parameter.

Certificate pricing

This section examines the pricing of a squared Constant Maturity Swap (CMS) certificate.
The payoff structure of these exotic coupons is replicated using CMS caplets, priced under the normal model in the forward swap measure, with their valuation given by:

$$V^{\text{CMS}}(0) = B(0,t) \cdot \left[\frac{C(K)}{\text{BPV}_0} + G'(R_s^0) \cdot \text{BPV}_0 \cdot \sigma^2 \cdot t \cdot \mathcal{N}\left(\frac{R_s^0 - K}{\sigma\sqrt{t}} \right) \right]$$

$$G(x) = \frac{x}{1 - \left(1 + \frac{x}{q}\right)^{-n}}$$

where $\mathcal{N}$ is the standard normal CDF, R_s^0 is the forward value of the swap rate that starts at each CMS reset and ends n year(s) later, q is the number of periods per year (in our case is equal to 2, since it is semiannual), t is the CMS caplet time to maturity year fraction, σ is the implied normal volatility of the model we want to use, C is the undiscounted price of European swaption payer that starts at each CMS reset date and ends n years later and BPV_0 is the forward basis point value of the underlying forward swap at reference date *0*.

We bound the non-linear payoff with a lower and an upper bounds constructed as piecewise linear approximations: the upper bound connects regularly spaced points $\{z_0 = -K, z_1^u, \ldots, z_N^u\}$ on the parabola, while a lower bound is formed by tangents at these points.

The time convention we used is the following business day convention, as asked, both for the payments of party A and B.

Construction of the upper and lower bound

In the following sections, we show how we constructed the upper and lower bounds for the payoff of a caplet, explaining our reasonments and all the computations we did to obtain this result.
The main idea is that both upper and lower bounds are considered, in a financial point of view, as portfolios of caplets with different strikes, each one with its notional, which corresponds to the slope of the part of the line.

Show that the point on interception of the straight lines in the lower bound are regularly spaced by Δz.

The description of the portfolio that reproduces the lower bound is explained in detail in the next section; here there is a theoretical proof of how in the lower bound the points of interceptions of the straight line we used to construct the lower bound are regularly spaced.
We prove this following an analytical approach, starting from the formula of the parabola $p(x) = 100 \cdot (x+K)^2$ (which represents the pay-off of a coupon) and the regular grid $\{z_0 = -K,\ z_1^u,\ z_2^u,\ \ldots,\ z_N^u\}$ spaced by Δz: we considered three general points of the grid $z_{i-1}^u = -K+(i-1)\cdot\Delta z$,$z_i^u = -K+i\cdot\Delta z$ and $z_{i+1}^u = -K+(i+1)\cdot\Delta z$ and then computed the value of the parabola in these points obtaining respectively:

$$p(z_{i-1}^u) = \left(-K + (i-1)\Delta z + K\right)^2 \cdot 100 = (i-1)^2 \Delta z^2 \cdot 100$$

$$p(z_i^u) = i^2 \cdot \Delta z^2 \cdot 100$$

$$p(z_{i+1}^u) = (i+1)^2 \Delta z^2 \cdot 100$$

At this point, we need the values of the slopes of the tangents to the parabola in the points $(z_{i-1}^u \ ; p(z_{i-1}^u))$, $(z_i^u \ ; p(z_i^u))$ and $(z_{i+1}^u \ ; p(z_i^u))$: these are computed thanks to the derivative of the function of the parabola $p'(x) = 100 \cdot 2 \cdot (x+K)$, finding:

$$p'\left(z_{i-1}^u\right) = 2 \cdot (-K + (i-1)\Delta z + K) \cdot 100 = 2 \cdot (i-1)\Delta z \cdot 100$$

$$p'\left(z_i^u\right) = 2 \cdot i\Delta z \cdot 100$$

$$p'\left(z_{i+1}^u\right) = 2 \cdot (i+1)\Delta z \cdot 100$$

We can now derive the analytical expression for each one of the tangents introduced above, starting from what we already know, the slope and the coordinates of a point where they pass through:

$$t_1(x) = (i-1)^2 \Delta z^2 + 2(i-1)\Delta z \left(x - (-K + (i-1)\Delta z)\right)$$

$$t_2(x) = i^2 \Delta z^2 + 2i\Delta z \left(x - (-K + i\Delta z)\right)$$

$$t_3(x) = (i+1)^2 \Delta z^2 + 2(i+1)\Delta z \left(x - (-K + (i+1)\Delta z)\right)$$

Now, it is sufficient to establish that the points of intersection between $t_1(x)$ and $t_2(x)$ (denoted as $P = (x_P, y_P)$) and between $t_2(x)$ and $t_3(x)$ (denoted as $Q = (x_Q, y_Q)$) are separated by Δz in the x-coordinate, to demonstrate that all such intersection points along the lower bound are uniformly spaced by Δz. This holds due to the generality of the points selected for the computations, so, if proved, we can conclude that all the points on interception of the straight lines in the lower bound are regularly spaced by Δz.

To prove it, we determine x_P and x_Q by solving the following systems of equations:

$$\begin{cases} y_P = t_1(x_P) \\ y_P = t_2(x_P) \end{cases} \quad \Rightarrow \quad t_1(x_P) = t_2(x_P) \quad \textit{and} \quad \begin{cases} y_Q = t_2(x_Q) \\ y_Q = t_3(x_Q) \end{cases} \quad \Rightarrow \quad t_2(x_Q) = t_3(x_Q)$$

The two equations yield, respectively, the following expressions for x_P and x_Q:

$$x_P = \frac{1}{2}(2i-1)\Delta z - K, \qquad x_Q = \frac{1}{2}(2i+1)\Delta z - K$$

Hence, the distance between points P and Q along the x-axis is:

$$|x_P - x_Q| = \left|\frac{1}{2}(2i-1)\Delta z - K - \left(\frac{1}{2}(2i+1)\Delta z - K\right)\right| = \left|-\frac{\Delta z}{2} - \frac{\Delta z}{2}\right| = \Delta z$$

Determine the X% for a fixed N, for the lower and upper bound using the two models' volatility smile

In order to find the unknown protection $P = X\%$, we have to consider the following relation between the two parts of the contract:

$$\text{NPV}_A = \text{NPV}_B^{wp} + (1 - P) \cdot B(t_0, t_N)$$

where NPV_A is the Net Present value of party A (what this party receives from the contract), NPV_B^{wp} is the one of party B but without protection and the last term is the protection that party A has to pay at maturity date, as defined in the contract.

To compute the value of the protection X%, this equation needs to be solved with respect to the value we want to deduce; so, our aim is to find an expression to calculate the Net Present Value of the two parties.

NPV_B^{wp} is what party B receives during the contract and it can be computed as:

$$\text{NPV}_B^{\text{wp}} = 1 - B(t_0, t_N) + c \cdot \sum_{i=1}^{N} \delta(t_{i-1}, t_i) \cdot B(t_0, t_i)$$

where B are the discount factors, δ is the relative *yearfrac* between a certain payment date and the previous one and c is the value of the coupon, equal to $\text{Euribor}_{6\text{m}} + 130$ bps.

The NPV of the coupons that party B pays to party A (NPV_A) is much more challenging to compute since these coupons are highly exotic, so it is not possible to find a closed formula for their NPV.
The easiest and more clever way to solve this problem is to find a portfolio whose pay-off is approximately equal to the one given by a single coupon.
We have built two positions: one portfolio of sup-replica of the non-linear pay-off and one of sub-replica. Both the upper and lower bounds consist in more long CMS caplet positions and one short CMS caplet position. Once given the discretization of the interval $[-K, z_N^u]$ for a fixed N, we found the notionals and strikes of all these caplets.

In the following we show the procedure followed:
Upper bound: we started by considering a caplet with strike equal to $-K$ and notional such that the slope of the linear part of the caplet pay-off matches the value of slope of the segment which connects the points of the parabola $(-K; p(-K))$ and $(-K + \Delta z; p(-K + \Delta z))$, where $p(x) = (x + K)^2 \cdot 100$ is the parabola representing the non-linaer payoff; this yields a notional of $100 \cdot \frac{(-K+\Delta z+K)^2-(-K+K)^2}{-K+\Delta z-(-K)} = 100 \cdot \frac{\Delta z^2}{\Delta z} = 100 \cdot \Delta z$.
Then, we need to add to our portfolio a second caplet with strike $-K + \Delta z$, whose notional increases the previous slope such that the total slope from $x = -K + \Delta z$ onward is equal to slope of the segment between the points $(-K + \Delta z; p(-K + \Delta z))$ and $(-K + 2\Delta z; p(-K + 2\Delta z))$; the total slope should be $100 \cdot \frac{(-K+2\Delta z+K)^2-(-K+\Delta z+K)^2}{-K+2\Delta z-(-K+\Delta z)} = 100 \cdot \frac{4\Delta z^2-\Delta z^2}{\Delta z} = 100 \cdot 3\Delta z$, so the notional of the second caplet must be $2\Delta z$.
Continuing with this procedure, we proved that the portfolio of sup-replica should be composed by N long caplets with strikes $\{z_0 = -K, z_1^u = -K + \Delta z, ..., z_{N-1}^u = -K + (N - 1) \cdot \Delta z\}$ and notional of $100 \cdot 2\Delta z$, except for the first one with notional of $100 \cdot \Delta z$; in addition to this, the portfolio includes a single short caplet with strike $z_N = -K + N \cdot \Delta z$ and notional of $100 \cdot (2N - 1) \cdot \Delta z$. We specify that the short caplet

is required in order to obtain a payoff capped at $M = 0.08$. The value of this portfolio is:

$$NPV_{ub} = 100 \cdot \Delta z \cdot Caplet(-K) + \sum_{i=1}^{N-1} 100 \cdot 2\Delta z \cdot Caplet(z_i^u) - 100 \cdot (2N-1) \cdot \Delta z \cdot Caplet(-K + N \cdot \Delta z) \quad (5)$$

Lower bound: we started by considering a caplet with strike equal to $-K + \frac{\Delta z}{2}$ and notional such that the slope of the linear part of the caplet pay-off matches the value of slope of the tangent to the parabola in the point $(-K + \Delta z\ ;\ p(-K + \Delta z))$, which is $p'(-K + \Delta z) = 100 \cdot 2 \cdot (-K + \Delta z + K) = 100 \cdot 2\Delta z$ (where $p'(x) = 100 \cdot 2 \cdot (x + K)$ is the derivative function of the parabola). Using the fact that the points on interception of the segments in the lower bound are regularly spaced by Δz, to obtain the slope of the tangent to the parabola in $(-K + 2\Delta z\ ;\ p(-K + 2\Delta z))$ we need to add to our portfolio a second caplet with strike $-K + \frac{3\Delta z}{2}$ whose notional increases the previous slope of the linear part of the first caplet to $p'(-K + 2\Delta z) = 100 \cdot 2 \cdot (-K + 2\Delta z + K) = 100 \cdot 4\Delta z$; this means that the notional of the new caplet must be $2\Delta z$. Continuing with this procedure, we demonstrated that the sub-replicating portfolio should be composed by N long caplets with strikes $\{z_1^l = -K + \frac{\Delta z}{2}, z_2^l = -K + \frac{3\Delta z}{2}, ..., z_N^l = -K + (N - \frac{1}{2}) \cdot \Delta z\}$ and notional of $100 \cdot 2\Delta z$; in addition, the portfolio includes a single short caplet with strike $z_N = -K + N \cdot \Delta z$ and notional of $100 \cdot (2N \cdot \Delta z)$ to cap the pay-off at $M = 0.08$. The value of this portfolio is :

$$NPV_{lb} = \sum_{i=1}^{N} 100 \cdot 2\Delta z \cdot Caplet(z_i^l) - 100 \cdot (N \cdot 2\Delta z) \cdot Caplet(-K + N \cdot \Delta z) \quad (6)$$

The upper and lower bounds converge to the parabola representing the payoff as N (the number of intervals in which we divide the grid) increases: the two boundaries get closer and closer as N gets higher, approximating always better the real payoff we cannot compute.
Once implemented the formulas of the NPV_{ub} and NPV_{lb} for a coupon on Matlab (`value_portfolio_suPreplica` and `value_portfolio_suBreplica` respectively), it is possible to show how the two NPVs converge to the same value as N increases:

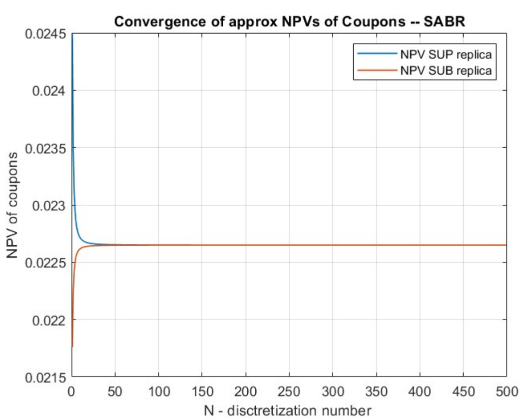


Figure 12: Convergence of upper and lower bounds NPVs - SABR model

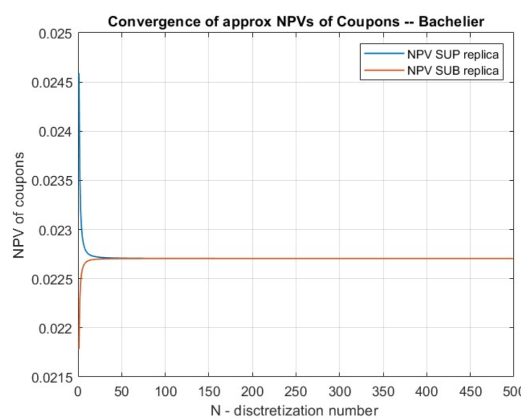


Figure 13: Convergence of upper and lower bounds NPVs - Additive Bachelier

So, if taken a value of N sufficiently big, it is possible to calculate a satisfactory value of the protection X%.

The value of X% is computed thanks to the Matlab function `Compute_protection`, which takes in input the total NPV of all coupons received by party A. The sum of the NPV of these coupons is obtained by using the function `Compute_NPV_coupons`: it takes in input the implied volatility surface with respect to which we want to make the computations of CMS caplet prices and the discretization number N; it gives as output the total NPV of the coupons both using the upper bound approximation of coupons pay-offs and the lower bound one.

In the following plot we show how the value of the protection X% changes as N varies, both for the upper and the lower bounds:

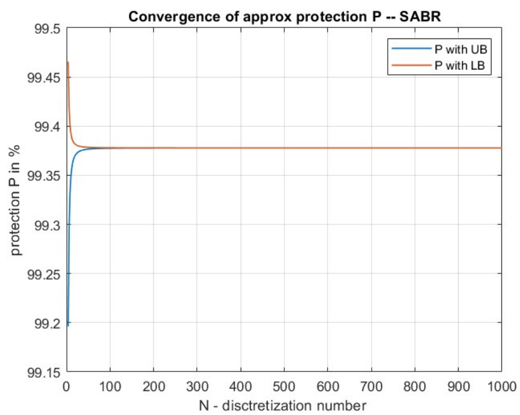


Figure 14: X% convergence with SABR implied volatilities

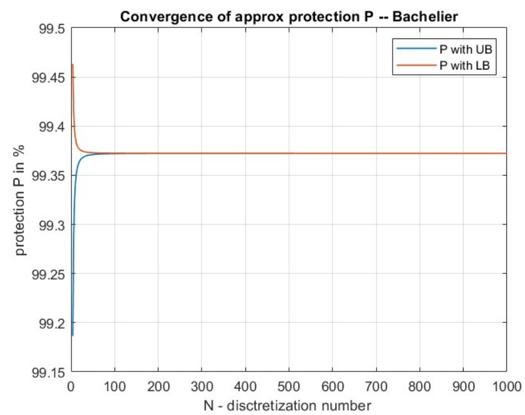


Figure 15: X% convergence with Addictive Bachelier implied volatilities

Since it is a protection, the behaviour of the two lines is opposite with respect to the NPVs due to the relation between the protection and the NPV of the coupons: as one increases, the other one decreases and viceversa. In fact, with the growth of N the NPV of the lower bound is increasing, while the corresponding protection is a monotone decreasing function.

Show that the error in the $X\%$ generated by the bounds decreases increasing N. Determine the $N^\star$ such that the difference between upper and lower bound is lower than a bp in absolute term in this case

In order to prove that the error in the computation of the protection decreases when increasing N, we calculated X% for different values of N both considering the upper bound ($X_{ub}\%$) and lower bound ($X_{lb}\%$), using the function `Compute_protection`; then, we compute the error as $|X_{ub}\% - X_{lb}\%|$ for each N.
In the following plot shows as the error is strictly decreasing and asymptotically converges to 0 as N increases:

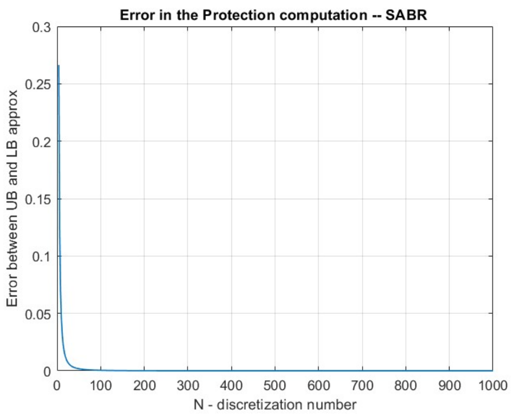


Figure 16: error $= |X_{ub}\% - X_{lb}\%|$ convergence with SABR implied volatilities

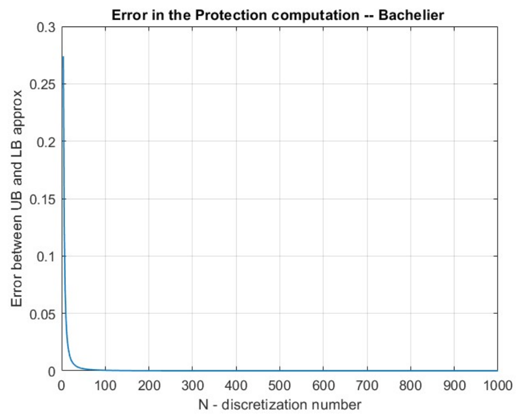


Figure 17: error$= |X_{ub}\% - X_{lb}\%|$ convergence with Addictive Bachelier implied volatilities

We found the value $N^{\star}$, considering both models, such that the error defined above is less than 1 bp and it corresponds to:

	$N^{\star}$
$SABR$	214
Additive Bachelier	214

The X% for this values of N is equal to:

	$X\%$
$SABR$	99.3778
Additive Bachelier	99.3723

Here we show how the error varies with respect to N: in the following plot in *loglog* scale, it is shown that it decreases as $\frac{1}{N^2}$:

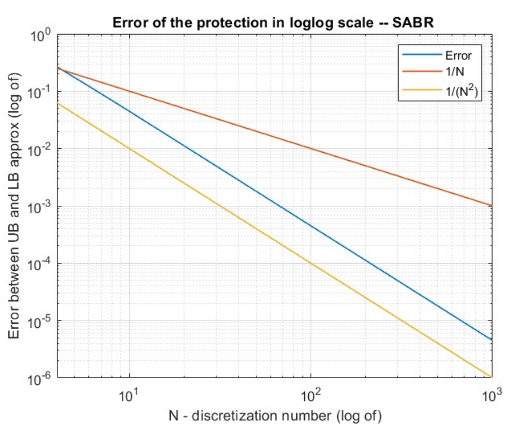


Figure 18: Loglog scale of the error - SABR model

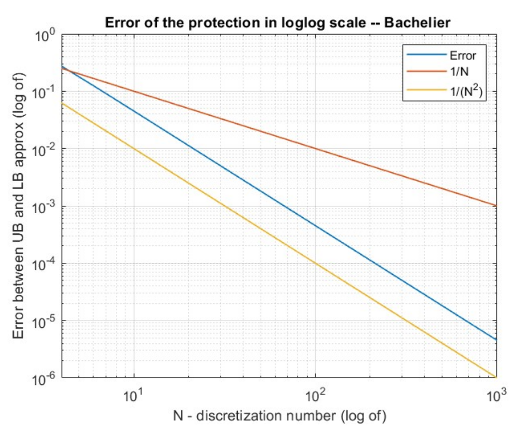


Figure 19: Loglog scale of the error - Additive Bachelier

Why has it been crucial to consider a model with an implied volatility smile to price this exotic product?

The pricing of this exotic product must account for the volatility smile rather than relying on a single implied volatility value. This necessity arises from the presence of multiple strike values—specifically, a vector of relative strikes, as shown in the Excel file *EURswptVols*. In this context, the Black model is inadequate, as it assumes a single strike level. The Black framework involves calibrating the model price to match the market quote and subsequently deriving the implied volatility by inverting the pricing formula for a fixed strike K. However, in the present case, the existence of a vector of strikes implies that each strike corresponds to a distinct volatility, as evidenced in the Excel data. Therefore, the assumption of a flat volatility surface across strikes is invalid and leads to inaccurate pricing.
The volatility smile is basically a curve that associates a certain volatility to a given strike K: the volatility σ is written as a function of K: $\sigma(K)$. To find a certain implied volatility, so a point of this smile, we fix the strike K and then match the market price with the model price in order to find the solution of this equation, which is the correct implied volatility (for the fixed strike K).

Moreover, the volatility smile is useful also in case of presence of digital risk: in this situation, it is necessary to consider in the pricing model both the Black term and the slope impact, which is the partial derivative of $\sigma(K)$ with respect to K, and this term is not negligible if compared to the previous one, so we need to add it in the formula.

Python code

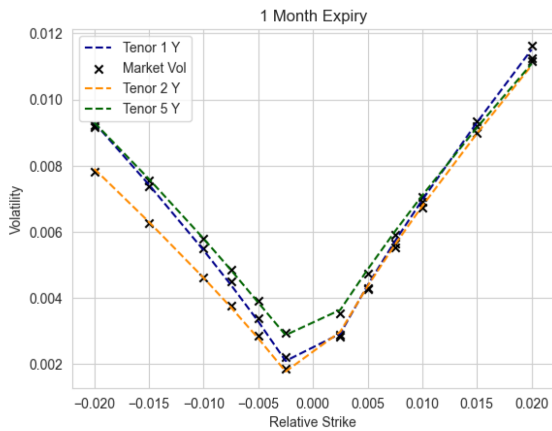
We also tried to implement all the codes also in Python language.

For the SABR model, the result obtained 20 are very similar to the one obtained in Matlab and the only considerable difference is in the running time of the code, both for the direct calibration and the cascade one, as it is shown in the following table:

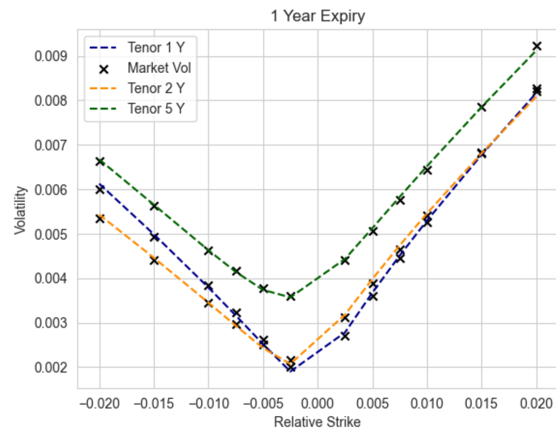
SABR model	Matlab	Python
Direct calibration	3.37s	12s
Cascade calibration	2.73s	0.7s

Table : Running time comparison of SABR model calibration in Matlab and Python

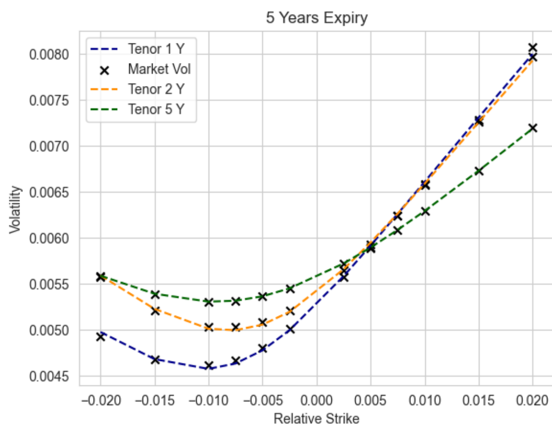
For the calibration, we used the function `minimize` from the `scipy.optimize` library. We experimented several methods of optimization including Gradient-based and Hessian-based algorithm. We found empirically that the most effective method for the cascade calibration was the `SLSQP` (Sequential Least Squares Programming) and for the direct calibration it was the `trust-constr`. These choices were based on a trade-off between convergence speed and the numerical result obtained.



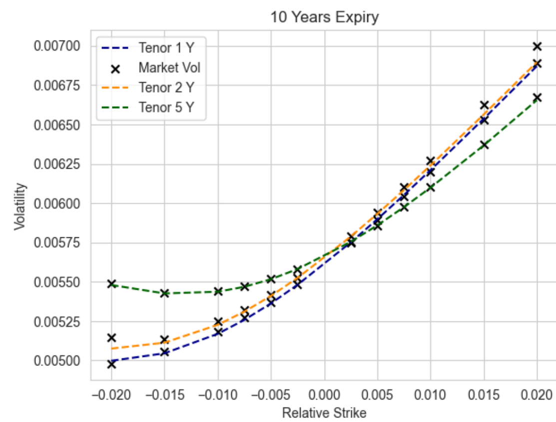
(a) Expiry 1m



(b) Expiry 1Y



(c) Expiry 5Y



(d) Expiry 10Y

Figure 20: Volatility smiles fixing expiry for Cascade Calibration

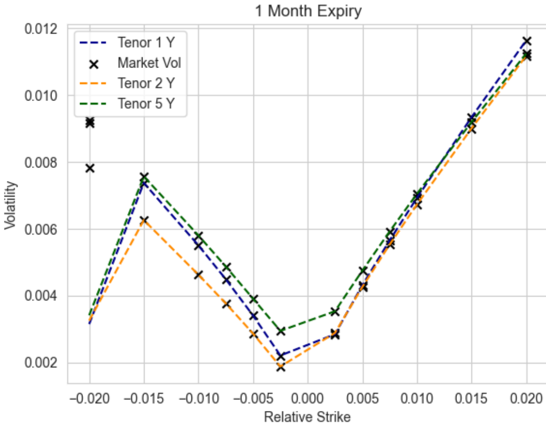
For the Additive Bachelier Model, the results obtained with the `quad` function of `scipy` were "unsatisfactory" (a important amount of computation time for a moderate-quality result). That's why we opted for an FFT approach, which was much faster and provided better results. The results were good overall, except for the lower relative strikes, where they tended to diverge from the market volatility 21.

Additive Bachelier	Matlab	Python
Direct calibration	2 min 23s*	9 min 16s
Cascade calibration	4 min 19s	2 min 58s

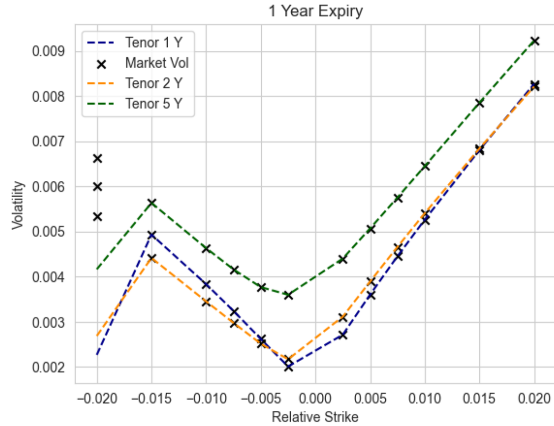
Table : Running time comparison of Additive Bachelier model calibration in Matlab and Python

Again we privileged a SLSQP optimization method which was the most efficient one.

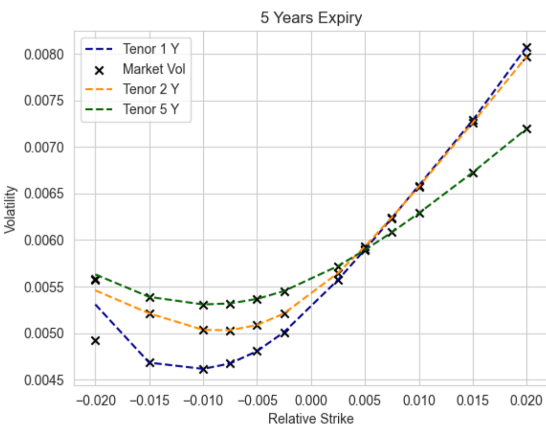
* : In MATLAB, the calibration time for the Additive Bachelier model is faster with the Direct method, as we tuned the initial conditions specifically for this method.



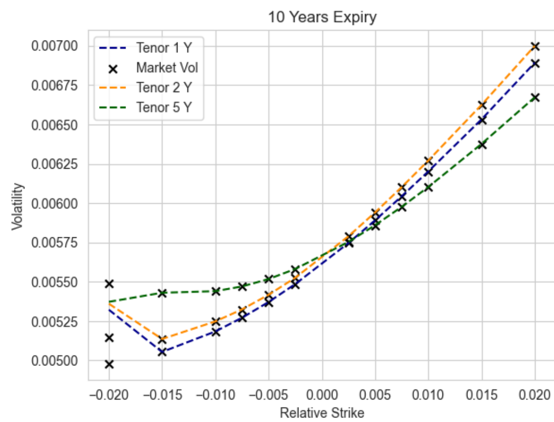
(a) Expiry 1m



(b) Expiry 1Y



(c) Expiry 5Y



(d) Expiry 10Y

Figure 21: Volatility smiles fixing expiry for Cascade Calibration with FFT

The certificate pricing was quite consistent with the Matalb result, however we obtained some "noise" for the upper bound 22. Although,we were unable to find the source of this perturbation, nevertheless it does not affect the final result.

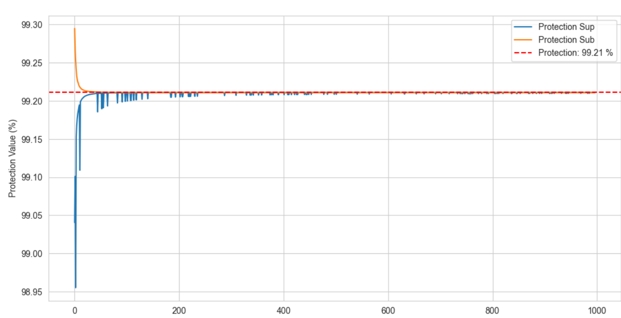


Figure 22: Protection (upper and lower bound)

Disclaimer: these results in Python were computed with the `tqdm` *librairy, in Matlab we used the* `tic toc` *instrument.*

Bibliography

Hagan, P. S., Kumar, D., Lesniewski, A. S., & Woodward, D. E. (2002). Managing smile risk. The Best of Wilmott, 1, 249-296.

Hull, J. C. (2009). Options futures and other derivatives. Pearson (7th edition).

Hagan, P. (2003). Convexity conundrums: Pricing CMS swaps, caps and floors. Wilmott Magazine 1, 38.

Baviera, R. (2022). A note on an additive model, 1-4.