BACHELOR'S THESIS / ENGINEERING 2025

Implied Volatility: the Bachelier Model

Eric Berenguer

upf. Universitat Pompeu Fabra Barcelona

Implied Volatility: the Bachelier Model

Option Pricing, Implied Volatility and Volatility Studies

TREBALL DE FI DE GRAU DE

Eric Berenguer Pons

Ralph Gregor Andrzejak, Elisa Alòs Alcalde

Grau en Enginyeria Matemàtica en Ciéncia de Dades

Curs 2024 - 2025

upf. Universitat Pompeu Fabra Barcelona

Escola d'Enginyeria

Dedicat als avis.

Abstract

The Bachelier model is often overlooked and counts with very limited literature, but at the same time, it is useful in rare extreme market events where some more mainstream models fail to capture market dynamics. Therefore, we would like to explore option pricing and implied volatility obtention under this model. In this thesis, we cover the development and implementation of option pricing with Monte Carlo methods that use the Bachelier model coupled with stochastic volatility models, which combined can reproduce volatility smile market dynamics. We have implemented methods to extract the implied volatility from option prices using the Bachelier model and numerical methods, as well as applying analytical approximations. Finally, we use the implied volatility values extracted to calibrate a SABR model to consolidate them into a unified framework. In this way, we establish an efficient method for option pricing and provide a methodology to link the Bachelier model to tools used by professionals in the financial industry.

Resum

El model de Bachelier és sovint ignorat i compta amb una literatura molt limitada, però al mateix temps és útil en situacions extremes al mercat, on models més convencionals fallen en capturar dinàmiques de mercat. És per això que volem explorar el *pricing* d'opcions i l'obtenció de la volatilitat implícita sota aquest model. En aquesta tesi cobrim el desenvolupament i implementació de *pricing* d'opcions amb mètodes de Monte Carlo que fan servir el model de Bachelier acoblat amb models de volatilitat estocàstica, que combinats són capaços de reproduir dinàmiques de mercat com la *smile* de volatilitat. Hem desenvolupat mètodes d'extracció de volatilitat implícita de preus d'opcions fent servir el model de Bachelier i mètodes numèrics, i també hem aplicat aproximacions analítiques. Finalment, hem fet servir els valors de volatilitat obtinguts per a calibrar el model SABR, consolidant les dinàmiques en un marc unificat. És així com hem establert un mètode eficient per valorar les opcions i hem proveït una metodologia que uneix el model de Bachelier amb eines pròpies de professionals a la indústria financera.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Options are one of the most utilized financial derivatives by professionals in the financial industry and retail investors alike. They are the basis of complex investment strategies and play a crucial role in the construction of hedging strategies [Daglish et al., 2007]. This is due to option contracts role as a leverage and having non-linear behavior of their returns.

The first mathematical framework that encompassed options was introduced as late (or as early) as 1900, by the then *normalien* mathematics PhD candidate Louis Bachelier. Bachelier founded the field of financial mathematics by proposing an arithmetic Brownian motion stochastic process as a model for an asset's price, but although being revolutionary and useful, this proposal was deemed flawed because it allows for prices to become negative. Later, other proposals for price dynamics were introduced, and the introduction of the famous Black-Scholes model, which modeled an option's underlying asset price, with a geometric Brownian motion, solved the "issue" of the possibility of negative prices.

The old Bachelier model was forgotten by industry professionals and fell out of use, but during the Great Financial Crisis of 2008, the then deemed impossible became real: traded instruments turned negative as interest rates and spreads fell below zero [Financial Times, 2009]. This suddenly made the imperfection of allowing negative prices of the Bachelier model a valuable characteristic that provided a mathematical framework to that twisted reality.

But this has not been a one-of-a-kind situation. In 2020, the COVID-19 pandemic halted all economic activity and forced the petroleum storage facilities to be full. Without space to store new produced oil and virtually no demand, the commodity became a liability and future contracts prices with an immediate delivery date turned negative [BBC News, 2020], creating yet another delicate situation in which the Bachelier model could be implemented.

This thesis provides an exploration of the Bachelier model, a discussion of variants the model has, as well as stochastic volatility models that when coupled

with the Bachelier model capture more complex market dynamics. Aside from a theoretical exploration, we also implement methodologies that allow the use of Bachelier's model in option pricing via Monte Carlo methods, and we develop a Conditional Monte Carlo method that improves the efficiency of the simulations with stochastic volatility models, which is an innovative approach.

Our ultimate goal is then to use the Bachelier model coupled with stochastic volatility as a framework to provide tools for managing option volatility risk. We show these model dynamics through volatility surfaces that contrast to the static and uniform volatility that the plain Bachelier model would produce (see section 2.4). Via the calibration of a constraint Stochastic Alpha-Beta-Rho (SABR) model, we then unify volatility dynamics with a global model that will operate under the same assumptions as the Bachelier model, thus producing real-world tools to be used in the financial industry with well-known properties [Hagan et al., 2002]. All of our methods and discussions are centered towards the vanilla European call option type.

Chapter 2

METHODS

2.1 The Bachelier-Type Models for Asset Prices

The Bachelier model, as presented in [Alòs and García Lorite, 2025], is defined by the following expression for the price of an underlying asset S_t, which is sometimes just called underlying, at time t:

$$dS_t = \sigma_t \left(\rho \, dW_t + \sqrt{1-\rho^2} \, dB_t \right), \quad t \in [0, T] \tag{2.1}$$

with: $T > 0$ is a rational number expressing the time to maturiry (or horizon) in years, W and B are two independent standard Brownian motions, $\rho \in [-1, 1]$ is a correlation factor that captures market dynamics by linking changes in asset prices to those of volatility and vice versa, and volatility is defined by σ_t, a square-integrable process adapted to the filtration[1] generated by W at time t.

When $\rho = 0$, the model is defined by a single Brownian Motion (BM) scaled by a factor of volatility:

$$dS_t = \sigma_t dB_t \tag{2.2}$$

When the volatility process is constant, we can drop the subindex of σ_t for σ. Models with $\rho = 0$ that furthermore have a constant volatility value are the classic Bachelier models, corresponding to the originally defined model in *Théorie de la Spéculation* [Bachelier, 1900]:

$$dS_t = \sigma dB_t \tag{2.3}$$

An important assumption of the Bachelier model is an assumed risk-free rate $r = 0$. This assumption simplifies the calculations while do not compromise on

[1] Filtrations in this context refers to the algebraic subobject, which for this purposes translates to all of the information provided by W at time t, essentially making σ related to W at time t but independent to previous to realizations of W.

the usability of the model for its main applications on forwards or interest rates, which their price is already discounted by their own nature.

2.1.1 Closed Pricing for Vanilla European Call Options

For the original Bachelier model (2.3), there is an analytical closed formula for the price for Vanilla European Call options. The derivation under this model for a fixed strike price k, and a time horizon (or time to maturity) T, comes from the random variable defined as follows:

$$C_T = (S_T - k)_+ \tag{2.4}$$

Where C_T is the value under the Bachelier model of the call option at time T, in essence the payoff function. S_T is the value of the underlying asset at T. The expression $(a)_+$ refers to the maximum of a and 0.

Therefore the value of the call option at time $t = 0$, C_0, can be defined as the expected value of C_T:

$$C_0 = E[C_T] = E[(S_T - k)_+] \tag{2.5}$$

Using an efficient market assumption, and knowing that the distribution of S_T under the classical Bachelier model (2.3) is Gaussian with mean S_0 (the value of the underlying when the option is conceived at time $t = 0$), and variance $\sigma^2 T$:

$$E[(S_T - k)_+] = \tag{2.6}$$

$$= \int_{k-S_0}^{\infty} (S_0 + x - k) \cdot \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{x^2}{2\sigma^2 T}\right) dx \tag{2.7}$$

$$= (S_0 - k)\Phi(d_{\text{Bac}}(k, \sigma)) + \varphi(d_{\text{Bac}}(k, \sigma))\sigma\sqrt{T} \tag{2.8}$$

with $d_{\text{Bac}}(k, \sigma) = \frac{S_0 - k}{\sigma\sqrt{T}}$, and φ as the probability density function (PDF) of a standard Normal distribution, and Φ as the cumulative density function (CDF) of a standard Normal distribution.

Derivation (2.7) was provided by Bachelier [Bachelier, 1900], while later it was adapted to the modern way of listing options, that being the common way of having fixed strikes and option prices determined by the market, leading to the final expression (2.8) [Schachermayer and Teichmann, 2008].

We will refer to the closed Schachermayer and Teichmann pricing formula (2.8) by the expression $\mathit{Bac}(T, S_0, k, \sigma)$.

2.2 Simulation of Prices Under the Bachelier Model

It is very common, especially in a financial context, to use Monte Carlo methods to find approximate solutions for models composed of complex mathematical expressions that have a random component. The methodology consists in simulating different possible outcomes the model might yield by transforming the expressions to a discrete domain, in case they are not already, and performing an iterative approach to retrieve possible outcomes. Once a large number of outcomes are generated, they are used to estimate the desired solution. For example, a common approach to finding the expected value of a stochastic expression is to just take the numerical average of the final results obtained by the random process simulations.

We have employed plain (raw) Monte Carlo methods as well as their variant conditional Monte Carlo methods to simulate option prices under the Bachelier model. This allows us to estimate option prices with stochastic volatilities, as well as with $\rho \neq 0$, corresponding to the general expression for the Bachelier model (see expression (2.1)), which is not supported by the closed pricing expression (2.8).

2.2.1 Raw Monte Carlo Method

For the first and most simple implementation, we have used raw Monte Carlo methods to simulate the price of the underlying asset at maturity (S_T). For this purposes, we have used a discretization of the underlying asset price using Euler's scheme.

The discretization of the model is done over a total time period T, and it is discretized in m steps. A total of n simulations are then performed. Each of the m steps $i+1$, with i ranging from 0 to $m-1$, of the simulated discretized process are modelled as follows:

$$\begin{aligned}
\Delta_i W &= W_{i+1} \cdot \sqrt{\frac{T}{m}} \\
\Delta_i B &= B_{i+1} \cdot \sqrt{\frac{T}{m}} \\
S_{i+1} &= S_i + \sigma_i \left(\rho \Delta_i W + \sqrt{1-\rho^2} \Delta_i B \right)
\end{aligned}$$

Here, W_{i+1} and B_{i+1} correspond to samples of two independent Standard Normal random distributions. σ_i is the i-th step of a stochastic volatility process driven by W. The volatility processes are simulated before the underlying simulations, and they will be discussed in their dedicated section.

When the m different steps have been simulated, an estimation of a possible price at maturity is produced (S_T^{MC}), each of the n independent simulations result will be called $S_{T,p}^{MC}$, where p ranges from 1 to n.

To retrieve the estimated option value at time 0 for a horizon T, the previous definition of $C_0 = E[(S_T - k)_+]$ is used and approximated by averaging the payoff function (2.4) of the n different simulations:

$$C_0^{MC} = \frac{1}{n} \sum_{p=1}^{n} (S_{T,p}^{MC} - k)_+ \tag{2.9}$$

2.2.2 Conditional Monte Carlo Method

Raw Monte Carlo methods are usually ineffective due to their high computing cost. In the particular application of the Raw Monte Carlo method described in subsection 2.2.1, the two simulated Brownian motions are sources of uncertainty. With the conditional Monte Carlo method, we are able to reduce the sources of uncertainty to a single one, meaning that only one Brownian motion will be simulated, allowing for faster computations and a reduction in variance of the simulation outputs.

The conditional Monte Carlo method developed in this thesis to achieve faster and more accurate simulations is based on transforming the general Bachelier model to the classical one by assuming that the Brownian motion W is known, this way it is possible to apply the closed formula to obtain the prices which allow us to avoid simulating *B* altogether.

If the model follows the classic Bachelier model (eq. 2.3), we know it exists a closed formula for the option's value (eq.2.8). First, given equation 2.2, if volatility is not constant and is defined by a stochastic process, we can define:

$$v_T = \sqrt{\frac{1}{T} \int_0^T \sigma_s^2 \, ds} \tag{2.10}$$

which allow us to transform the model (2.2) into the classical Bachelier model (2.3), with v_T as a constant volatility parameter in the closed formula. Following the reasoning in equation 2.5, we can define the expected value of the payoff function to be equal to the expected value of the closed formula with v_t as its volatility parameter:

$$E[(S_T - k)_+] = E[\text{Bac}(T, S_0, k, v_T)] \tag{2.11}$$

Furthermore, since the Brownian motion W is the driver of the stochastic volatility process, we can make use of the conditional probabilities by assuming

W is known:

$$E[\text{Bac}(T, S_0, k, v_T)] = E[E[\text{Bac}(T, S_0, K, v_T) \mid W]]$$

Now for $\rho \neq 0$, we need to use the expression of S_T in terms of the general model definition (2.1):

$$S_T = S_0 + \rho \int_0^T \sigma_s \, dW_s + \sqrt{1-\rho^2} \int_0^T \sigma_s \, dB_s$$

Gathering all of the pieces, conditioned on W, S_T can be seen as a process with deterministic volatility given by:

$$\sqrt{1-\rho^2} \cdot v_T$$

and an initial asset price:

$$S'_0 = S_0 + \rho \int_0^T \sigma_s \, dW_s$$

Allowing us to transform it to the classic model with $S_0 = S'_0$ and $\sigma = \sqrt{1-\rho^2} \cdot v_T$, thus concluding:

$$E[E[\text{Bac}(T, S_0, k, v_T) \mid W]] = E[\text{Bac}(T, S'_0, k, \sqrt{1-\rho^2} \cdot v_T)].$$

Simulations under the conditional framework are now possible with the general model (2.1), giving us the freedom to use $\rho \neq 0$ and stochastic volatilities and still being able to use the closed formula to avoid simulating the Brownian motion B (2.1) and erasing the uncertainty that simulating it produces.

To compute the option price with this method, a total of n simulations are performed. For each simulation, m samples of a standard normal distribution are taken to get the known values of W_p, the Brownian motion W for the p-th simulation, which will then be used to simulate the m steps of a stochastic volatility process. Then $v_{T\,p}$ and $S'_{0\,p}$, the values of v_T and S_T given W_p, are computed with a Euler scheme to finally use the closed formula and get the result of $E[\text{Bac}(T, S'_{0\,p}, k, v_{T\,p}) \mid W_p]$. The procedure with m newly sampled values for W is repeated n times, and the results are averaged to obtain an approximation $C_0^{Cond\,MC}$ for $E[E[\text{Bac}(T, S_T, k, \nu_T) \mid W]]$, the vanilla European call option price (2.11):

$$C_0^{Cond\,MC} = \frac{1}{n} \sum_{p=1}^{n} \text{Bac}(T, S'_{0\,p}, k, \sqrt{1-\rho^2} \cdot v_{T\,p})$$

2.2.3 Monte Carlo Methods - Implementation Considerations

For Monte Carlo methods to be effective, the number of simulations needs to be as large as possible to be more accurate, and at the same time, the step size has to be as small as possible to better approach a continuous behavior, which makes Monte Carlo methods computationally intensive. Furthermore, data structures have to be thoughtfully managed in order to minimize the loading and unloading of data into memory, which can even be a more important factor in improving the simulations execution time than the computations themselves.

We have implemented our Monte Carlo methods in Python within a Windows environment, and therefore the discussions will be centered in an efficient implementation in a setup of our characteristics. Although Python is rich in usability and libraries that notably streamline implementations and provides the developer with valuable development tools, it is not optimal for efficient memory management. With a few optimization steps the run time can be drastically reduced, and we will discuss a few considerations that achieve a notable reduction on the simulation computation times when being run on Python.

Memory Usage and Efficient Computation

Each of the n different simulations are independent and can be treated on their own and by linearly conducting separate simulations n times, each simulation with their m discretized steps. However, The single simulation iteration approach is not recommended, as libraries like *numpy* have thoughtfully developed subroutines on a lower level that optimize matrix operations. With an approach where a matrix of size $n \times m$ is created and each of the m different steps across n simulations are produced simultaneously, the subroutines optimize data loading and computations, resulting in a vastly reduced execution time, as it is demonstrated in table 2.1.

Simulations Parallelization

Since all the simulations are independent from each other, in a multicore environment it is useful to divide the different n simulations into equal chunks among all available cores, so that we can use the subroutines that efficiently deal with matrix operations at each core in parallel, further decreasing the time to compute all simulations.

To achieve parallelization, we make use of *joblib*, a library developed for parallel execution of Python code. Our implementation divides the total number of simulations through a parallel executor with as many worker processes as specified. Our implementation counts with seven parallel processes.

An important remark is that *joblib* uses the *loky* backend module, which in Windows environments produces tremendous overheads because it relies on the *spawn()* function to create the children processes [Joblib Developers, 2025]. This means that each child process does not share memory with the parent and all of the data is copied in full every time a new process is created or loaded. To increase efficiency using the *joblib* library for parallelization, the simulations must be performed in a script as lightweight as possible, which includes only the essential parts for the execution and storage of the simulations and that imports the minimum amount of libraries, so as to reduce the overhead loading time as much as possible.

2.2.4 Monte Carlo Implementations Performance Analysis

In the later section 3.1, we use Monte Carlo methods to simulate prices with 30 different times to maturity and 40 strike prices for each maturity; and for each simulated price we use $n = 180$ simulations with $m = 1500$ individual time steps each, which yields a total of 3240000000 ($3.24{\cdot}10^9$) simulated steps, contextualizing how a small improvement in efficiency in each of the simulated steps can add up to a notable improvement.

To compare the efficiency of the considerations presented in section 2.2.3, we performed the simulation for a single price 12 times with each method and compared the results. Since 12 is a 1% of the total price simulations required for each *volatility surface* (see section 3.1), it gives an intuition on how much time the execution of prices simulations for a whole surface takes with each method.

All the executions presented in table 2.1 have been performed in individual, separate scripts, with $m = 1500$ simulated steps and $n = 1800$ simulations, and with the same parameters:

- Bachelier Model Parameters:
  $\rho = 0.3$, $k = 698.5$, $S_0 = 700$, and $T = 5$

- Stochastic Volatility model[2] parameters:
  $\kappa = 5$, $\theta = 0.46$, $\nu = 0.46$, and $\sigma_0 = 0.46$.

[2]Heston model is used, see see subsection 2.3.1 for further information on the model and its parameters

Method	Average Price	Prices Variance	Time (seconds)
Serial Raw Monte Carlo	1.505466	2.226164	385.94
Serial Conditional Monte Carlo	1.501937	0.000271	262.24
Matricial Raw Monte Carlo	1.494563	2.189593	4.36
Matricial Conditional Monte Carlo	1.501923	0.000263	5.28
Parallel Raw Monte Carlo	1.500225	2.247322	1.71
Parallel Conditional Monte Carlo	1.502191	0.000192	2.34

Table 2.1: Different Monte Carlo implementations with the same parameters.

Implementations Comparison

It is evident in the comparison table (Table 2.1) how poor memory management by serially executing the code (one step at a time for each simulation) is inefficient. By simply exploiting the numpy subroutines by modifying the code to execute it as a matrix, in our test case the execution time is reduced from 385.94 to 4.36 seconds, a reduction of 98.8% (for the raw Monte Carlo method).

If, in addition, the code is divided and executed in parallel, the execution time is cut one more time, from 4.36 to 1.71, providing a total reduction of 99.56% over the execution time of the serial implementation (of the raw Monte Carlo method), which corresponds to just implementing the formulas directly and not using the considerations presented in section 2.2.3. By taking only 0.5% of the time needed for a straightforward implementation, we are able to try a wider variety of parameters with our optimized code.

Conditional Monte Carlo efficiency

In table 2.1, it is observable how the prices simulated using the conditional Monte Carlo have variance 4 orders of magnitude smaller compared to those simulated with the raw method. This makes a clear case of the conditional implementation being more precise than the raw one, and even though the execution time is greater for the conditional method with the same number of simulations as the raw method, the drastic reduction in variance would allow us to use a much smaller number of conditional simulations in practical cases, compared to raw Monte Carlo methods, to achieve our desired level of precision, making the conditional method more efficient in an applied scenario. This early convergence of the conditional Monte Carlo method can be seen in figure 2.1.

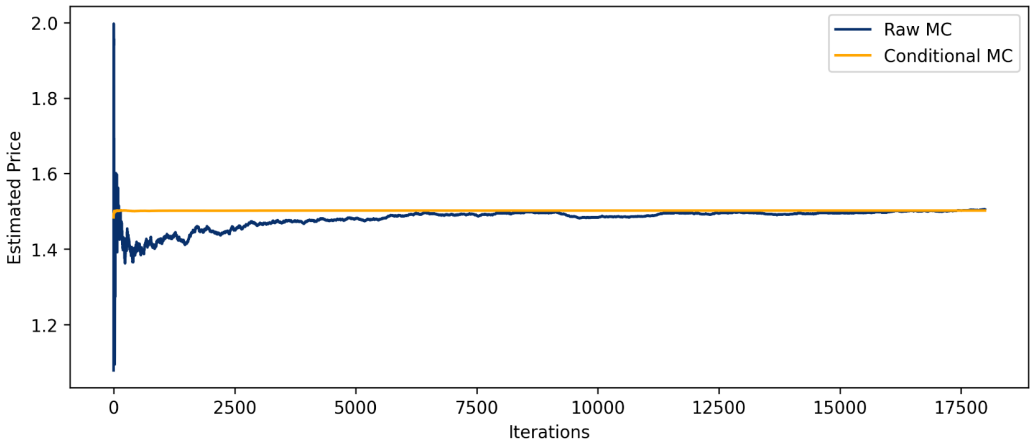


Figure 2.1: Convergence Plot of different Monte Carlo methods.

2.3 Simulation of Volatility Paths

Here we present two stochastic volatility models (Heston and Log-Normal) we use to simulate the stochastic variances for the general Bachelier model (expression 2.1), which innovates on the classic model by allowing to better capture market dynamics, as well as an overview of their implementation. We also employ the concepts used for the Monte Carlo simulations presented in section 2.2.3 to efficiently simulate the volatility paths.

We have chosen both models for their widespread use in the professional financial industry as well as their ability to capture volatility clustering, a phenomenon where high volatility values tend to be concentrated in determined periods of time. And we have chosen two instead of one to compare the results yielded by models with different volatility behaviors, as the Heston is more rough, meaning that it captures irregular and non-smooth volatility paths, while the log-normal is more stable over time. At the same time both models count with identical parameters that make them very customizable and interesting to experiment with.

We would like to remark that all the uncertainty sources in each of the following stochastic volatility models being applied to our Bachelier simulations come from the Brownian motion W as seen in equation 2.1.

2.3.1 Heston Model

The Heston model is a well-known model that is capable of modeling both stochastic volatilities and asset prices. As an extension of the well-known Black-Scholes model, it is also log-normal in nature, but its variance process presents some in-

teresting properties, like being Chi-Square distributed, making it possible to adapt well to volatility smiles and suitable for rough volatilities.

The model can be described by this coupled stochastic differential equations (SDE):

$$\frac{dS_t}{S_t} = \sqrt{V_t} dZ_t \tag{2.12}$$

$$dV_t = \kappa(\theta - V_t)\, dt + \nu\sqrt{V_t}\, dW_t \tag{2.13}$$

with V_t as the instantaneous variance, or equally, the quadratic variance of $\frac{dS_t}{S_t}$, or σ_t^2; and Z_t and W_t being correlated Brownian motions.

The model parameters can be used to calibrate the model to market data quite effectively. This is possible because each has a concrete, defined, and independent effect on the properties of the volatility process:

- $\kappa > 0$: mean reversion speed — in essence, how fast the variance process reverts to its long-term mean θ,
- $\theta > 0$: long-term mean level — the value that V process will tend to revert to,
- $\nu > 0$: volatility of variance.

Even if the model proposes a complete framework that includes asset prices, only the part that models the quadratic variance path V will be used. The implementation we followed is the one presented in [Andersen, 2007], a discrete form simulation using an Euler scheme with m steps over a total time period of T (years):

$$V_{i+1} = V_i + \kappa(\theta - (V_i)_+)\frac{T}{m} + \nu\sqrt{(V_i)_+} \cdot W_i \cdot \sqrt{\frac{T}{m}}, \quad i \text{ an integer} \in [0, m]$$

Notice that this scheme might produce negative values of V_i, which can be handled in different ways, but we will set after the simulation each V_i to $(V_i)_+$, the maximum between 0 and V_i, since it is the method that produces the minimum amount of bias resulting from a simulated discrete process [Andersen, 2007].

Once the process of V has been simulated, since V is the squared instantaneous volatility, we will retrieve our values of interest (σ_i), which in the discrete domain is: $\sigma_i = \sqrt{V_i}$.

2.3.2 Log-Normal Model

Although similar to the Heston model for variance, this model slightly differs from the Heston in different aspects. The distribution of σ_t, as the name of the model suggests, corresponds to a Log-Normal distribution in this case.

The model for the logarithm of the volatility can be defined as follows:

$$d\log(\sigma_t) = \kappa\left(\log(\theta) - \log(\sigma_t)\right)dt + \nu\, dW_t \tag{2.14}$$

where the parameters κ, θ, and ν play the same role as in the Heston model, W_t is a Brownian motion and σ_t is the volatility.

The main differences between the Heston and Log-Normal models are in the behavior and properties of the process. In the Heston model, ν is scaled by $\sqrt{V_t}$, making it more volatile at extreme values of variance, while it is not scaled in the Log-Normal model. Also, the Log-Normal model is always positive by construction. The simulation implementation for the Log-Normal model uses an Euler scheme of identical construction to the Heston model adapted to the formula 2.14.

2.4 Implied volatility Obtention

Implied volatility, sometimes abbreviated as IV, in the Bachelier model is defined by the only volatility parameter that given an option with time to maturity T, strike k and underlying price at time of inception S_0, can yield the option's market traded value C^{Market} under the Bachelier model, or equivalently the value σ_{Bac} such that $\text{Bac}(T, S_0, k, \sigma_{Bac}) = C^{Market}$.

Notice that the original Bachelier model assumes the volatility to be constant, and therefore if real market prices follow this model, we would expect the volatility to be constant among all strikes and times to maturity. However, as is widely known and also reported in [Alòs and García Lorite, 2025], options traded in real markets do not follow this kind of behavior, but instead a phenomenon called *volatility smile* emerges from the extraction of implied volatilities of real market prices, making the volatility values appear on a curve in the shape of a smile across different strike prices, or interchangeably for options with the same time to maturity but different moneyness, with *moneyness* being the ratio of the initial asset price over the strike price (*moneyness* $=\frac{S_0}{k}$).

The volatility smile is a phenomenon that is essential to manage risk and appropriately price various financial derivatives and can provide valuable information about the sentiment of the market. The causes of the smile are reported to be related to market momentum and transaction costs that more extreme market events might generate [Peña et al., 1999], but the smile also reflects future market expectations for the returns of the underlying asset.

For the obtention of the volatility smile or skew, option prices with severally different strike prices spaced evenly but with the same time to maturity should be extracted. The prices can be obtained directly from the market by obtaining their traded listed price, or, in our case, by simulating the price with methods like Monte Carlo simulations. This will allow us to see the volatility dynamics, and we can observe in figure 2.2 that the classic Bachelier model is unable to capture the volatility smile.

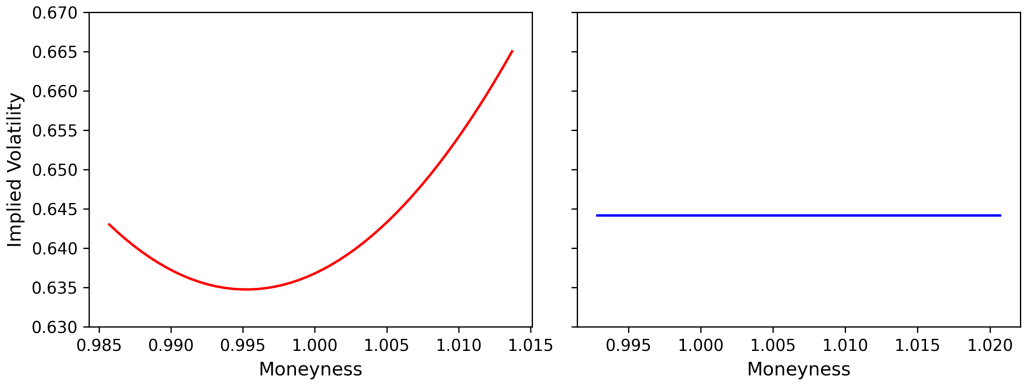


Figure 2.2: Left: Volatility smile example (own elaboration and obtained through a SABR model calibrated with simulated stochastic volatility prices); Right: Implied volatility of prices simulated under a classic Bachelier model.

2.4.1 At-The-Money Implied Volatility

At-The-Money (ATM) options are options at which the strike price is equal to the underlying price at time $t = 0$ (or equivalently with $moneyness = 1$). Using the closed formula 2.8, they can be priced with $\text{Bac}(T, S_0, S_0, \sigma_{Bac})$, notice that $d_{\text{Bac}}(S_0, \sigma)$ is 0, and with simple arithmetic on the pricing formula, the price of an ATM call option C_0^{ATM} results in an easily invertible equation that allows us to directly retrieve the implied volatility for this particular case without using more complex methods [Schachermayer and Teichmann, 2008]:

$$C_0^{ATM} = \sigma_{Bac}\sqrt{\frac{T}{2\pi}} \Rightarrow \sigma_{Bac} = C_0^{ATM}\sqrt{\frac{2\pi}{T}} \tag{2.15}$$

Thanks to this simple derivation, it is possible to analytically retrieve the implied volatility of any option when the strike and the price of the underlying are the same allowing some computations to be streamlined, and also to use the implied volatility value computed with 2.15 as a sanity check to contrast results provided by other methods.

2.4.2 Implied Volatility Extraction with Numerical Methods

To find the parameter σ_{Bac} for our simulated options with price C^{Market}, each with their defined k, T, and same S_0; we can define an optimization problem of a loss function consisting of the squared difference of the price yielded for different volatility parameters and the market price:

$$\sigma_{Bac} = Min_{\sigma_{Bac}}[(C^{Market} - \text{Bac}(T, S_0, k, \sigma_{Bac}))^2] \tag{2.16}$$

To perform this root-finding task, we have used Brent's algorithm. Obtaining the derivative of the expression 2.16 would be very complex and not feasible, and Brent's algorithm guarantees to find the root of the function and converges faster than most other algorithms to find roots without derivatives [Brent, 1971], and the algorithm can be implemented with the Python library $Scipy$. The usage of this method is possible since we can pin the root of the expression to be around previously observed parameters or sensible ones: market volatility on even the most volatile assets rarely surpass a threshold of 1, and by definition it cannot be negative. We, therefore, have defined a bracket of search between $1{\cdot}10^{-6}$ and 30, two almost impossibly extreme volatility parameters that at the same time also delimit a domain small enough to search for the root efficiently.

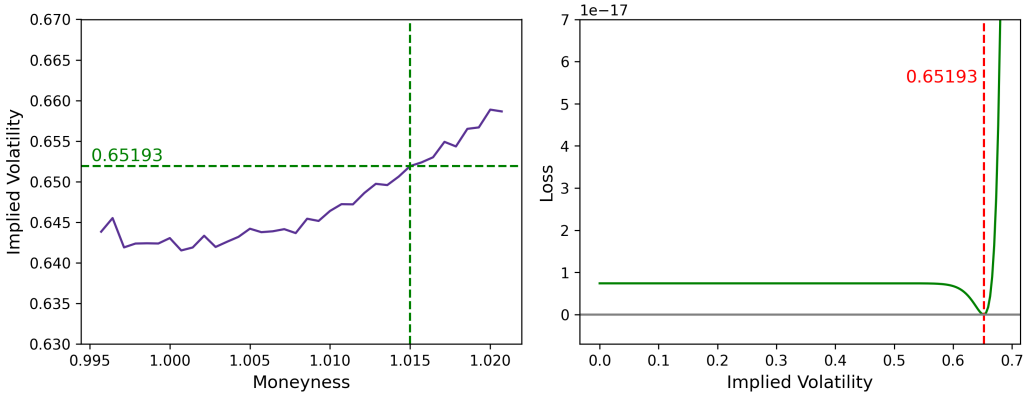


Figure 2.3: Left: volatility smile obtained with Brent's method, point for $moneyness = 1.015$ highlighted with the corresponding implied volatility value; Right: Loss function (2.16) for $moneyness = 1.015$, root highlighted in red.

Brent's method is usually effective enough, but there are some important considerations when applying it. The values around the root of the loss function tend to get close to zero, see that in figure 2.3 right the values around the root are of magnitude 10^{-17}, and the function can have a small slope in certain areas. These properties of the loss function make it hard to treat it with a computer, since the

decimal points of precision within python libraries may not be entirely accurate, specially when dealing with *float* variables, which may lead to cumulative errors in rounding on the order of 10^{-17} [Python Software Foundation, 2025].

With some Bachelier prices, the computer sensitivity is not good enough to register changes in price with different values of σ_{Bac}, which is not solvable if the implementation requires the usage of libraries that use standard data types. An example of the sensitivity issue can be seen in figure 2.4.

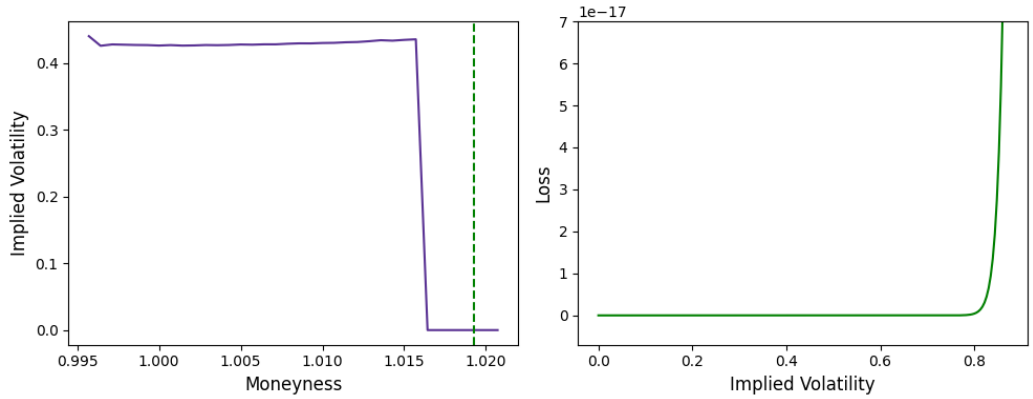


Figure 2.4: Left: volatility smile obtained with Brent's method, point with erroneous IV highlighted; Right: Loss function (2.16) for the selected point.

2.4.3 Implied Volatility Extraction with an Analytical Approximation

The literature on implied volatility for the Bachelier model is quite limited, but [Choi et al., 2022] provides a numerical approximation to analytically extract the implied volatility of a classic Bachelier model. The method presented in [Choi et al., 2022] is defined by the following computations:

$$\sigma_{Bac} = \sqrt{\frac{\pi}{2T}} \left(2C^{Market} - (S_0 - k)\right) h(\eta) \tag{2.17}$$

$$\text{for} \quad \eta = \frac{v}{\operatorname{atanh}(v)} \quad \text{and} \quad v = \frac{|S_0 - k|}{2C^{Market} - (S_0 - k)} \tag{2.18}$$

$$\text{and} \quad h(\eta) \approx \sqrt{\eta} \cdot \frac{\sum_{k=0}^{7} a_k \eta^k}{1 + \sum_{k=1}^{9} b_k \eta^k}, \tag{2.19}$$

with the presented $h(\eta)$ being a Chebyshev approximation. This is due to the fact in the derivation of this method, Choi et al. consider options with $moneyness \neq 1$

to be perturbations of the ATM case, and therefore they modeled this behavior with an approximation of $h(\eta)$, whose coefficients are defined as:

$$
\begin{aligned}
a_0 &= 3.99496\,16873\,45134 \times 10^{-1} & b_1 &= 4.99053\,41535\,89422 \times 10^{1} \\
a_1 &= 2.10096\,07950\,68497 \times 10^{1} & b_2 &= 3.09357\,39367\,43112 \times 10^{1} \\
a_2 &= 4.98034\,02178\,55084 \times 10^{1} & b_3 &= 1.49510\,50083\,10999 \times 10^{3} \\
a_3 &= 5.98876\,11026\,90991 \times 10^{2} & b_4 &= 1.32361\,45378\,99738 \times 10^{3} \\
a_4 &= 1.84848\,96954\,37094 \times 10^{3} & b_5 &= 1.59891\,96976\,79745 \times 10^{4} \\
a_5 &= 6.10632\,24078\,67059 \times 10^{3} & b_6 &= 2.39200\,88917\,20782 \times 10^{4} \\
a_6 &= 2.49341\,52853\,49361 \times 10^{4} & b_7 &= 3.60881\,71083\,75034 \times 10^{4} \\
a_7 &= 1.26645\,80513\,48246 \times 10^{4} & b_8 &= -2.06771\,94864\,00926 \times 10^{2} \\
 & & b_9 &= 1.17424\,05993\,06013 \times 10^{1}
\end{aligned}
$$

This method improves on the previously presented in [Choi et al., 2009] with the mention that for values of v close enough to zero, η should be calculated with its Taylor expansion:

$$\eta = \frac{1}{1 + \frac{v^2}{3} + \frac{v^4}{5} + \cdots}$$

Although for ATM options we can skip the approximation altogether and use the formula 2.15 instead.

[Choi et al., 2022] is the only analytical approximation for the volatility of an arithmetic Brownian motion process, and the method is reported to have an error of magnitude of the order of 10^{-13}, which is suitable for most professional uses in the financial industry. However, we have encountered some challenges during the implementation phase that limited the usability of this method in our case.

The main complication we found stems from the fact that Choi's approximation method relies heavily on calculating extremely accurately all of the different terms. An example of a challenging computation is $\operatorname{atanh}(v)$, since $\lim_{v\to 1^-} \operatorname{atanh}(v) = +\infty$ and $\lim_{v\to -1^+} \operatorname{atanh}(v) = -\infty$, and most values of v are extremely close to 1 or -1, it is essential to obtain as many decimal places as possible so that the calculation is accurate enough.

In our implementation, we try to solve the lack of precision using only calculations performed with variables of the $decimal$ data type, which had been manually set at 150 decimal places, a significant increment over the 28 points of precision that $decimal$ variables have at standard. We also use the equivalence of $\operatorname{atanh}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$ to improve its accuracy, since most built-in methods to compute $\operatorname{atanh}(x)$ are not accurate enough, and we have tested different thresholds to use the Taylor expansion of η instead of the summation of the term products, as it is not defined in [Choi et al., 2022] what counts as a small enough value of v to use Taylor's expansion, all to make our computations as precise as possible.

When the adaptations are implemented improving the computing accuracy, some volatility values stabilized for on options with $k > S_0$, giving results similar to the ones obtained with Brent's method and that corresponded to values that could be expected. Nevertheless, volatility values remain unstable for options with $k < S_0$ and for some values close to the ATM region. A visual representation of the unstable values produced by Choi's method due to computational inaccuracies can be observed in figure 2.5.

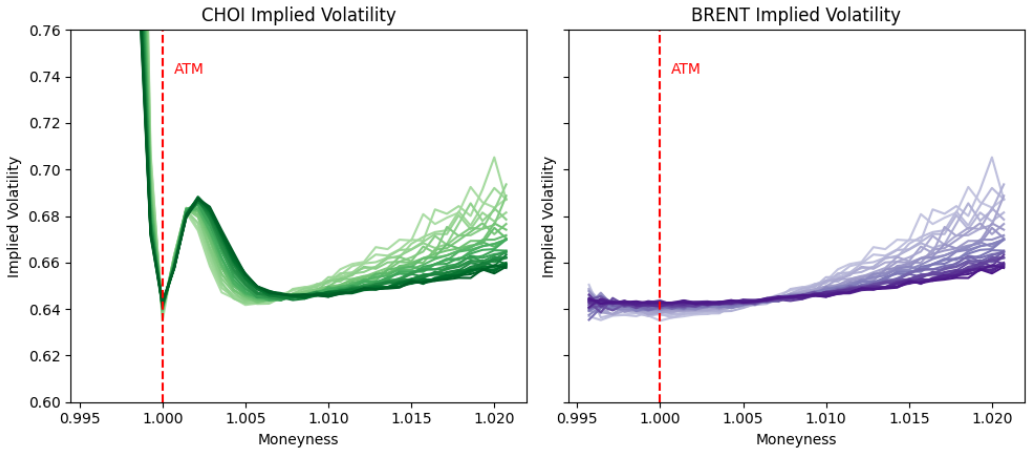


Figure 2.5: Volatility Smiles obtained with our implementations of Choi and Brent methods, each curve represents the smile for a different time to maturity, the color deepens as the time to maturity increases, which ranges from 4 to 9 years.

The persistent computational inaccuracies of our Choi's method implementations prevent us from further using this method, and we therefore favor Brent's method to retrieve the implied volatility values. Nevertheless, the proven accuracy of Choi's method would make a feasible implementation of it quite valuable for an accurate estimation of implied volatilities values while also being able to work with a known range of accuracy for the estimation. No code is provided by Choi and there is no successful, or unsuccessful, implementation of this method in the public domain, to the best of our understanding.

Chapter 3

RESULTS AND APPLICATION

3.1 Volatility Surface

Volatility smiles provide information on the implied volatility of options with the same time to maturity but different strikes, and volatility surfaces provide the same information but contextualized with an additional axis for different times to maturity. In essence, a volatility surface consists of an infinite number of volatility smiles, one for each time to maturity that can ever exist, plotted alongside one another forming a 3-dimensional object.

Volatility surfaces are a widely used tool in trading desks around the world and are great for showing how a model is able to reproduce smile dynamics, since it also contextualizes volatility information over different maturities [Daglish et al., 2007]. As a reminder, the classic Bachelier model (expression 2.3) would produce a completely flat volatility surface, as it assumes a constant volatility parameter, but volatility surfaces in reality are not flat and therefore it is interesting to be able to produce models that adjust to this reality.

To reproduce volatility surfaces, we have simulated prices in a mesh of 40 evenly distributed strike prices (with strike prices ranging from 695 to 715, and $S_0 = 700$) by 30 different times to maturity (ranging from 4 to 9 years). To extract the implied volatility from the simulated prices, we have used Brent's method.

3.1.1 Rendered Volatility Surfaces

In figure 3.1, we can see how both volatility models, Heston and Log-Normal, work in different scales, and even if they have the same parameters, the scale of the obtained implied volatilities of their simulated prices are on two different levels and exhibit different behavior, which is to be expected.

With the different volatility surfaces we obtained and the different proper-

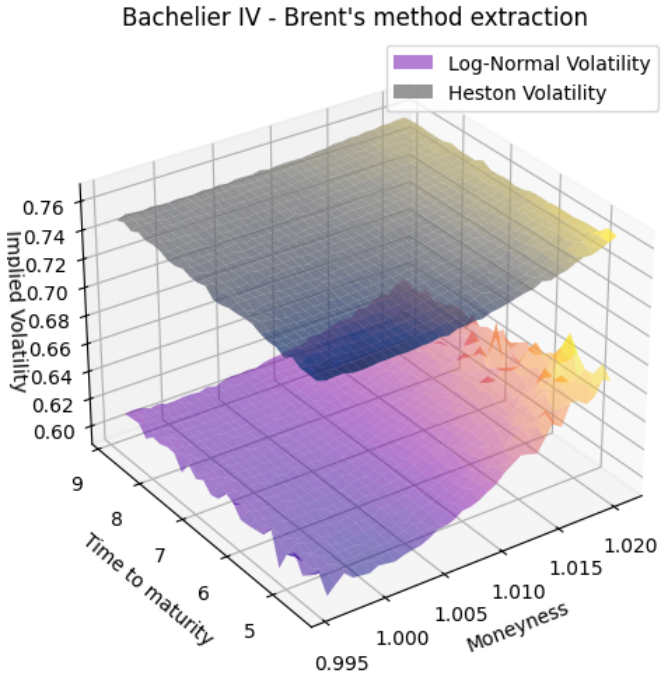


Figure 3.1: Volatility surface, parameters: $\rho = 0.3$, $\kappa = 5$, $\theta = 0.8$, $\nu = 0.3$.

ties they exhibit, we showcase and demonstrate with a practical example that the Bachelier-type models coupled with a stochastic volatility model are perfectly capable of capturing the volatility smile phenomenon and complex structures that may arise in the market prices, an approach that we did not find in the literature available to us.

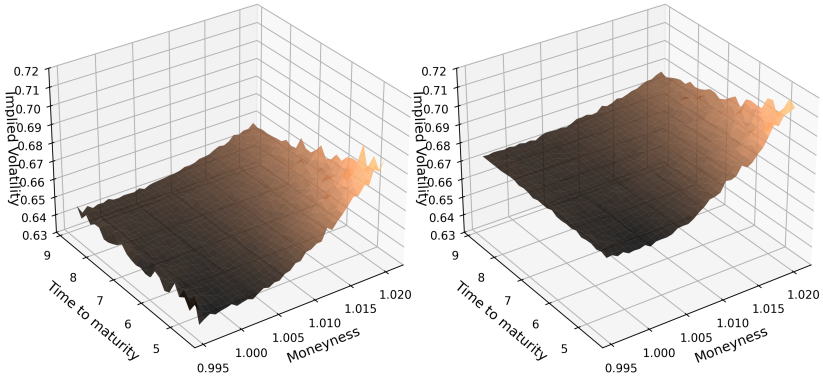


Figure 3.2: Volatility surfaces, with parameters: $\kappa = 5$, $\theta = 0.46$, $\nu = 0.6$; right with $\rho = 0.3$, and left with $\rho = 0$, Heston volatility model.

In the two simulations included in figure 3.2 (both using the same Heston volatility model and parameters), we can compare the effect that ρ as a confirmation exercise for the dynamics of the general expression 2.1. As can be seen, the surface for the model with $\rho = 0.3$, has a different shape for the volatility surface, as the volatility in the Bachelier model when $\rho = 0$ is correlated with the asset path, and also produces different values overall, which falls within the expected dynamics.

To observe more effects that the ρ and stochastic volatility model parameters have on the volatility surfaces, more plots resulting from simulations with different parameters are available in the appendix (section A.3).

3.2 The SABR model, an implied volatility real-world application

As shown in the resulting volatility surfaces in section 3.1, we are able to produce option prices that capture the volatility smile, and we are also able to retrieve the implied volatility for each combination of T and k, but each implied volatility value obtained would correspond to a different classic Bachelier model (2.3). There is a 'one-to-one relation between the price of a European Call option and the value σ_{Bac}' (adapted from [Hagan et al., 2002]). The SABR model is capable of correctly capturing smile dynamics on its own, with a single set of parameters. The model also counts with closed algebraic formulas that allow the retrieval of the European call option price, as well as the implied volatility, as a function of S_0 and k.

3.2.1 SABR model definition

The SABR model presented in [Hagan et al., 2002] has the following definition:

$$dS_t = \sigma_t S_t^{\beta}\, dW_1, \tag{3.1}$$

$$d\sigma_t = \nu \sigma_t\, dW_2, \tag{3.2}$$

$$dW_1\, dW_2 = \rho\, dt \tag{3.3}$$

Where $\beta \in [0, 1]$ is an exponent of S_t, and W_1 and W_2 are two Brownian motions correlated by a factor of ρ (which can be seen in 3.3).

In the SABR model, β plays an important role in defining the behavior of S_t, since if $\beta = 1$, it would be of log-normal nature, if $\beta = \frac{1}{2}$ it would then be of Cox-Ingersoll-Ross nature. But what is relevant for us is that for $\beta = 0$ it follows an arithmetic Brownian motion [Hagan et al., 2002], and notice how,

indeed, the equation 3.1 of the SABR model transforms into the equation for the classic Bachelier model with stochastic volatility (equation 2.2) for the case $\beta = 0$.

3.2.2 Hagan's Implied Volatility Formula

Hagan provides a closed formula that can express the implied volatility of an option, if the parameters σ_0, ν, and β are known and for a fixed T, as a function of S_0 and k:

$$\sigma_{Bac}(k, S_0) = \frac{\sigma_0}{(S_0 k)^{(1-\beta)/2}\left\{1 + \frac{(1-\beta)^2}{24}\log^2\left(\frac{S_0}{k}\right) + \frac{(1-\beta)^4}{1920}\log^4\left(\frac{S_0}{k}\right) + \cdots\right\}} \cdot \left(\frac{z}{x(z)}\right) \cdot \left\{1 + \left[\frac{(1-\beta)^2}{24}\cdot\frac{\alpha^2}{(S_0k)^{1-\beta}} + \frac{1}{4}\cdot\frac{\rho\beta\nu\sigma_o}{(S_0k)^{(1-\beta)/2}} + \frac{2-3\rho^2}{24}\nu^2\right]T + \cdots\right\} \tag{3.4}$$

where

$$z = \frac{\nu}{\sigma_0}(S_0k)^{(1-\beta)/2}\log\left(\frac{S_0}{k}\right), \quad x(z) = \log\left(\frac{\sqrt{1-2\rho z + z^2} + z - \rho}{1-\rho}\right)$$

The part of the expansion omitted in the formula 3.4 (denoted by ...), is too small to have any real effect on the result and therefore it is not included [Hagan et al., 2002]. But even without the inclusion of the extra terms, the formula is quite heavy and not too stable in our implementations. But in the original paper [Hagan et al., 2002], a simplification of the formula for values of k close enough to S_0 is provided (without specifying a benchmark on when it is considered close enough):

$$\begin{aligned}\sigma_B(k, S_0) = \frac{\alpha}{S_0^{1-\beta}}\Bigg\{1 - \frac{1}{2}(1-\beta-\rho\lambda)\log\frac{k}{S_0} + \\ + \frac{1}{12}\left[(1-\beta)^2 + (2-3\rho^2)\lambda^2\right]\log^2\frac{k}{S_0} + \cdots\end{aligned} \tag{3.5}$$

Where the ratio:

$$\lambda = \frac{\nu}{\alpha}f^{1-\beta} \tag{3.6}$$

3.2.3 SABR model calibration

An important concept, especially in a professional setting, is that of model calibration, which is a process where a model's parameters are optimized to fit observed market properties. As the SABR model provides a single model for the whole volatility smile, it is useful for professionals who wish to have a single integrated model to manage various risk exposures to calibrate a SABR model periodically [Hagan et al., 2002].

SABR calibration - Bachelier implied volatility procedure

The SABR model is not applicable to options with different T, as its parameters are not time sensitive. It still remains a useful tool that encompasses the whole smile dynamics, but to reproduce a volatility surface, it has to be fitted to every possible time to maturity.

The process we followed to construct a volatility surface with the SABR model consists of the following steps:

- First, we extract the implied volatility values for a surface as described in section 3.1, in our case we use the prices simulated with conditional Monte Carlo methods.
- Secondly, we use a nonlinear approximator to find the optimal α, ρ, and ν SABR parameters that adjust to the data by using Hagan's formula (expression 3.5) with fixed $\beta = 0$, since we are interested in the arithmetic Brownian model that is a proxy for the Bachelier model. We have used the reduced volatility formula 3.5 since we could not find convergence on the extended formula 3.4.
- Lastly, we use the calibrated model parameters to retrieve the corresponding volatility values to reconstruct the surface.

For the non-linear least squares problem that needs to be solved to find the optimal SABR parameters, we have used the Levenberg–Marquardt algorithm, which is capable of dealing with this problem [Marquardt, 1963], and has a straightforward implementation with Python using the library *Scipy.Optimize*.

Convergence using this method is not always guaranteed, and for random initializations, we found it often does not converge. After comparing with other SABR calibration literature, like [Fernández et al., 2013], we have chosen initial parameters similar to their optimized parameters: $\alpha = 0.15$, $\rho = -0.5$, and $\nu = 0.5$, which allow for convergence. Our estimated parameters did differ significantly from those, as seen in table 3.1, but the mean relative error of the implied volatility of our SABR estimations is 0.5610^{-2}, which is in the same range as that

in [Fernández et al., 2013]. Since the parameter estimations had a sensible variance of: 0.0778, 1.6710^{-4}, and 0.3275 for α, ρ, and ν respectively, the difference in the parameters may be explained due to the imposed constraint of $\beta = 0$.

Maturity (T)	α	ρ	ν
4.83	447.2888	$-3.1073 \cdot 10^{-2}$	-20.5788
7.83	449.0173	$-9.0711 \cdot 10^{-2}$	-11.6263

Table 3.1: Estimated parameters for two SABR calibration results, fixed $\beta = 0$.

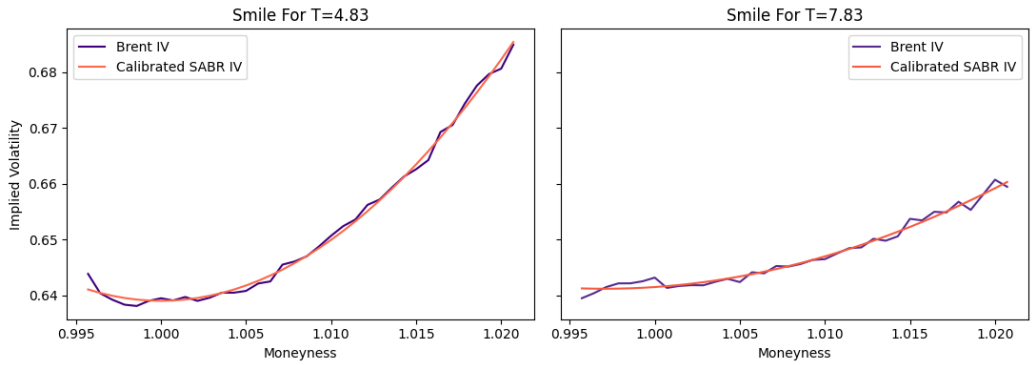


Figure 3.3: SABR IV calibrated with the IV obtained with Brent's method, both contrasted (SABR parameters correspond to those on table 3.1)

In table 3.1, we can observe the different parameters that we have obtained during the calibration process, and in figure 3.3 we can see the parameters of implied volatility obtained through Brent's method used to calibrate the SABR model, and also the volatility smile that the calibrated model produces.

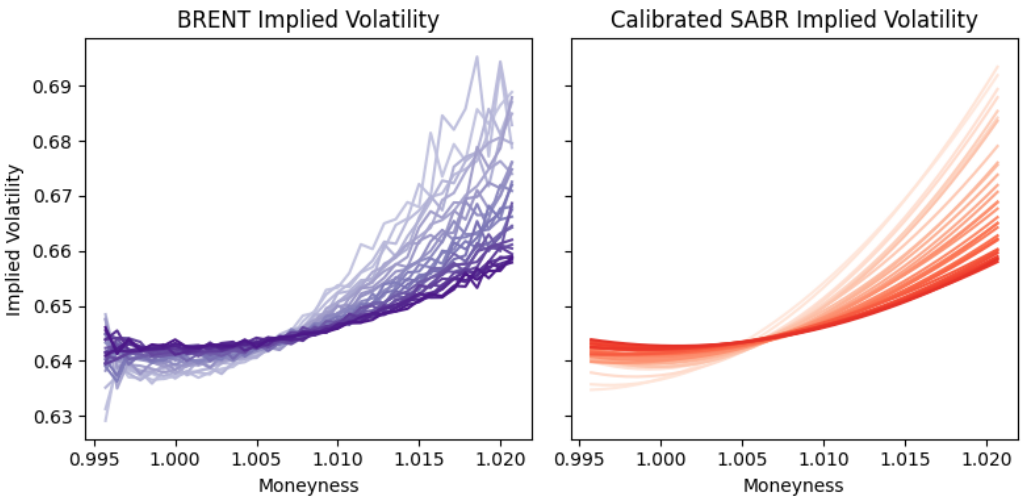


Figure 3.4: Volatility "smiles" obtained with Brent's method (left), and the resulting ones from our model calibration (right). Each curve represents the smile for a different time to maturity, the color deepens as the time to maturity increases, which ranges from 4 to 9 years.

With the mesh of strike prices and times to maturity we use for our volatility surfaces computation in section 3.1, we can then reconstruct the surfaces by calibrating a SABR model for each time to maturity, and then using the calibrated parameters to obtain the SABR implied volatility. Figure 3.4 shows the different smiles obtained form the calibration, and in the figures below (figures 3.6 and 3.5), the reconstructed surface using the calibrated models can be observed in contrast to the one obtained through our implementation of Brent's method.

The calibrated SABR parameters obtained from our implied volatility surfaces have a direct and valuable application in a real-world scenario, since we can introduce the hypothesis of a Bachelier model paired with stochastic volatility to real-market data and use this information to calibrate a SABR model. This proposition allows managing volatility smile risks under a unified model and maintaining the characteristics of a Bachelier model.

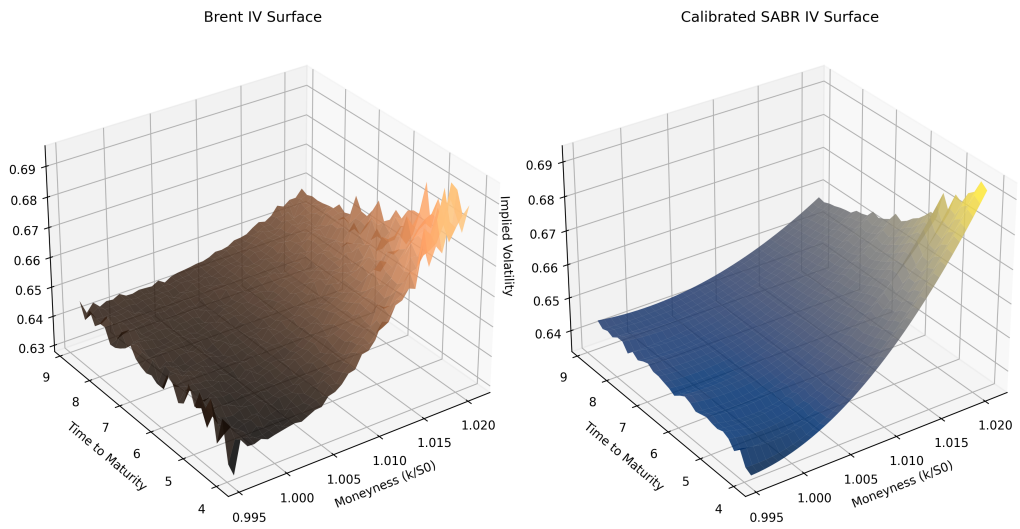


Figure 3.5: Brent (left), and calibrated SABR (right) volatility surfaces

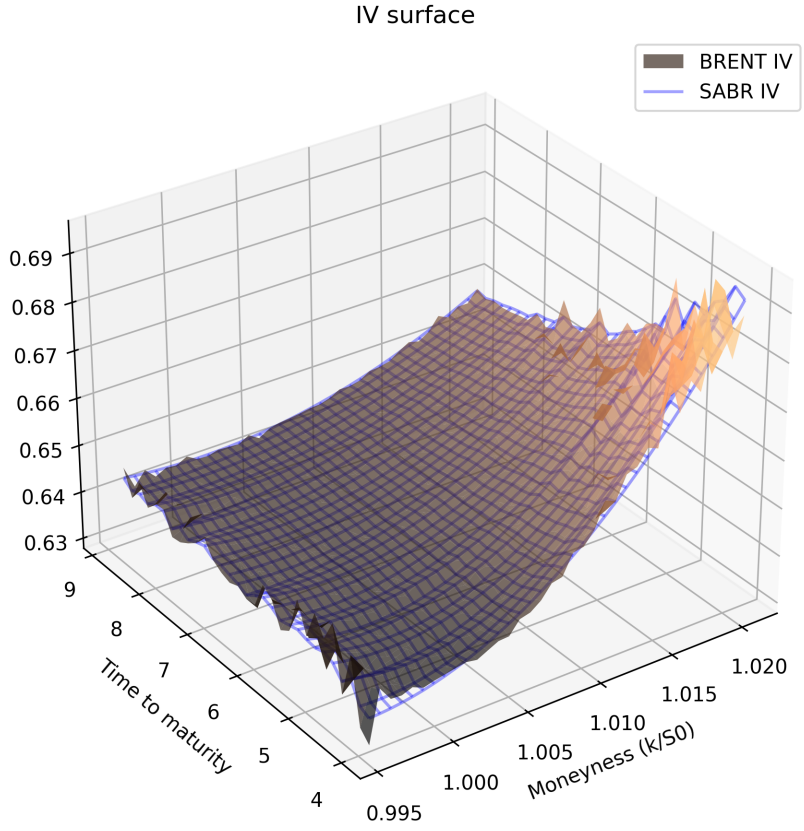


Figure 3.6: Brent and calibrated SABR volatility surfaces, superimposed

Chapter 4

CONCLUSIONS

The Bachelier model still has some niche applications in finance, where some situations require a model where asset prices can adopt negative prices. The exclusive ability of the model, apart from the equivalence of the SABR with $\beta = 0$, to deal with these situations makes it truly relevant, and therefore it cannot be neglected or forgotten. The lack of extensive literature covering the Bachelier model, especially in contrast to the literature on the most famous models like the Black-Scholes-Merton model, makes it ever so hard to be able to overcome challenges in the implementation phase or to find interesting extensive information of practical applications of the model.

Despite the challenges in finding reliable information and virtually no examples of applications covering our topics, we were able to show that smile dynamics can be attributable to a Bachelier model paired with a stochastic volatility model, and we showed it through the development of a pricing simulation Conditional Monte Carlo algorithm that is a clear improvement over the application of a raw Monte Carlo implementation.

Furthermore, we were able to extract the implied volatility under the Bachelier model with numerical methods, and discussed the usability of analytical approximations for the implied volatility under the Bachelier model. Our pursuits allowed us to use the well-known SABR model, essential in managing risk in the financial industry, showing compatibility of the Bachelier model with modern tools to manage volatility risk.

But although the overall successful results of our implementations, we urge to investigate more to find a robust and consistent method to extract the implied volatility under the Bachelier model. In periods of high uncertainty, like these last months, having a model that can deal with the most unexpected scenarios is all-important, and additional research on the Bachelier model should be a priority if it is to be used for sensible tasks in a professional setting without failures.

Bibliography

[Alòs and García Lorite, 2025] Alòs, E. and García Lorite, D. (2025). *Malliavin Calculus in Finance: Theory and Practice*. Chapman and Hall/CRC, 2 edition.

[Andersen, 2007] Andersen, L. B. G. (2007). Efficient simulation of the heston stochastic volatility model. Technical report, Banc of America Securities. Available at `http://www.ressources-actuarielles.net/EXT/ISFA/1226.nsf/769998e0a65ea348c1257052003eb94f/1826b88b152e65a7c12574b000347c74/\$FILE/LeifAndersenHeston.pdf`.

[Bachelier, 1900] Bachelier, L. (1900). *Théorie de la Spéculation*, volume 17. Annales Scientifiques de l'École Normale Supérieure, Paris.

[BBC News, 2020] BBC News (2020). Us oil prices turn negative as demand dries up.

[Brent, 1971] Brent, R. P. (1971). An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14(4):422–425.

[Choi et al., 2009] Choi, J., Kim, K., and Kwak, M. (2009). Numerical approximation of the implied volatility under arithmetic brownian motion. *Applied Mathematical Finance*, 16(3):–. Originally posted June 4, 2007; revised March 16, 2010.

[Choi et al., 2022] Choi, J., Kwak, M., Tee, C. W., and Wang, Y. (2022). A black–scholes user's guide to the bachelier model. *Journal of Futures Markets*, 42(5):959–980.

[Daglish et al., 2007] Daglish, T., Hull, J., and Suo, W. (2007). Volatility surfaces: Theory, rules of thumb, and empirical evidence. *Quantitative Finance*, 7(5):507–524.

[Fernández et al., 2013] Fernández, J. L., Ferreiro, A. M., García-Rodríguez, J. A., Leitao, A., López-Salas, J. G., and Vázquez, C. (2013). Static and dynamic sabr stochastic volatility models: Calibration and option pricing using gpus. *Mathematics and Computers in Simulation*, 94:55–75.

[Financial Times, 2009] Financial Times (2009). Negative 30-year rate swap spread linger.

[Hagan et al., 2002] Hagan, P. S., Kumar, D., Lesniewski, A. S., and Woodward, D. E. (2002). Managing smile risk. *Wilmott Magazine*, pages 84–108.

[Joblib Developers, 2025] Joblib Developers (2025). *Joblib Documentation: Serialization & Processes*. `https://joblib.readthedocs.io/en/latest/parallel.html\#serialization-processes`.

[Marquardt, 1963] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441.

[Peña et al., 1999] Peña, I., Rubio, G., and Serna, G. (1999). Why do we smile? on the determinants of the implied volatility function. *Journal of Banking & Finance*, 23(8):1151–1179.

[Python Software Foundation, 2025] Python Software Foundation (2025). Floating point arithmetic. Python 3 documentation.

[Schachermayer and Teichmann, 2008] Schachermayer, W. and Teichmann, J. (2008). How close are the option pricing formulas of bachelier and black-merton-scholes? *Mathematical Finance*, 18(1):155–170.

Appendix A

SUPPORTING MATERIAL

A.1 Code

All the scripts utilized for the Monte Carlo price simulations and other implementations, a general notebook containing the code used for volatility analysis and supporting material elaboration, and the results in csv files for the simulated prices used in the notebook, can be found in the following GitHub:

`https://github.com/eberpo/Implied_Vol_TFG/tree/main`

A.2 Glossary

In this section certain general financial concepts that are given as known in the document are briefly explained.

- **Financial Derivative**: Any kind of financial contract that it's value is derived from an underlying asset.
- **Option**: Options in their simplest constitution (called vanilla European options) are contracts that give the option, but not the obligation, to buy or sell an underlying asset at a specified price, called the strike price or strike for short, at a specified time in the future called maturity. The underlying asset can be anything, from equities (stock from companies) and bonds, to commodities, or even other derivatives. Options that give the option to buy are called call options, and those that give the right to sell are called put options.
- **Future**: Futures, sometimes called forwards, are contracts that give the obligation to buy an underlying asset at a specified time, called the delivery date.

When the asset is delivered, if it is a bond, a stock or any other form of intangible asset, it is simply transferred to the future contract holder. However, it is important to note that for tangible underlying assets, such as gold or petroleum, there can be a possibility that the delivery is done in real life, meaning that the future contract holder would have the obligation pick and store the actual physical underlying asset. Physical delivery of an underlying asset is an uncommon occurrence, but it is important to consider it.

- **Hedging**: A strategy in which by buying/selling certain financial instruments, has the objective to limit the downside risks of a given position in the market.

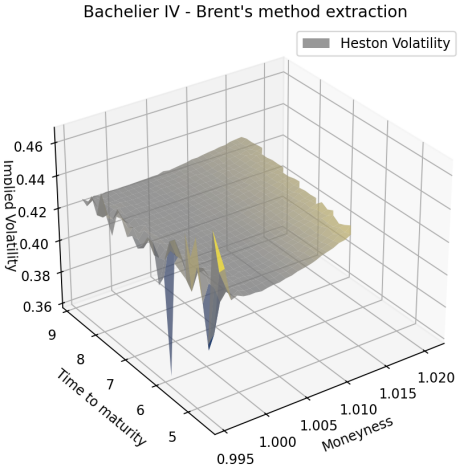
- **Traded/Quoted Price**: The last price for which a certain financial instrument was sold in the market.

- **Risk-Free Rate**: In a financial context, the monetary value in the future is diluted (or discounted) by the Risk-Free rate, which is the rate of return that an investment with no risk has. The Risk-Free rate is usually derived from the rate of return that the United States Treasury bonds have, since it is ”impossible” for the United States government to default on its debt it is considered a risk 0 investment.

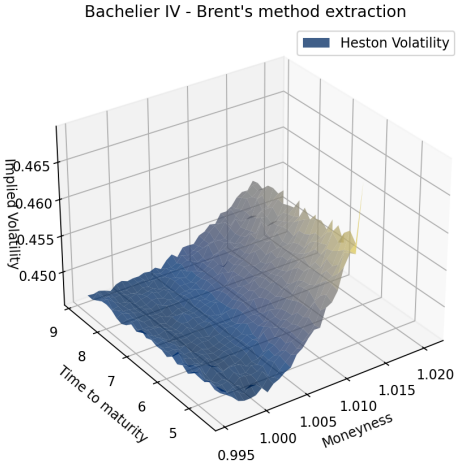
- **Efficient Market Hypothesis**: This is a hypothesis that states that the current price of an asset includes of all the information available, making it impossible to predict future changes in price with past information, introducing the notion that markets move randomly and in a way that does not allow anyone to make a systematic profit out of it without assuming any risk.

- **Spread**: The spread is defined by a difference between two assets, it can refer to the difference of price when buying or selling an asset, or the difference in return by two different assets, usually the additional return of an asset against the risk-free rate.

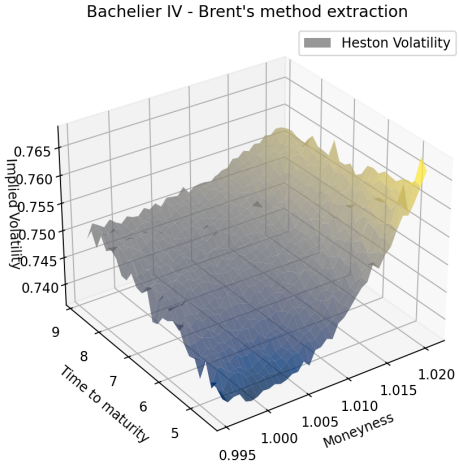
A.3 Additional Implied volatility Surface Visualizations



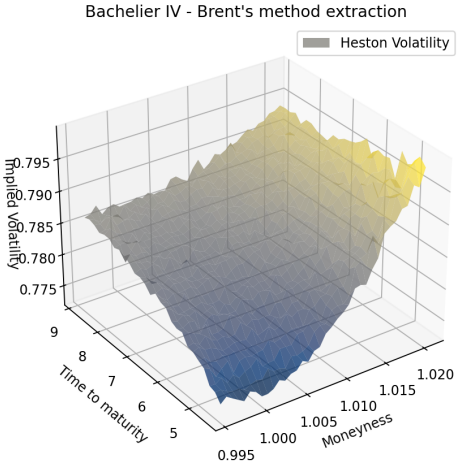
((a)) Bachelier with $\rho = 0.3$, $\kappa = 5$, $\theta = 0.2$, $\nu = 0.3$



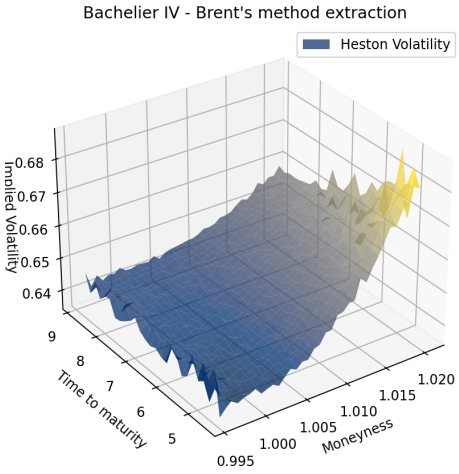
((b)) Bachelier with $\rho = 0$, $\kappa = 5$, $\theta = 0.2$, $\nu = 0.3$



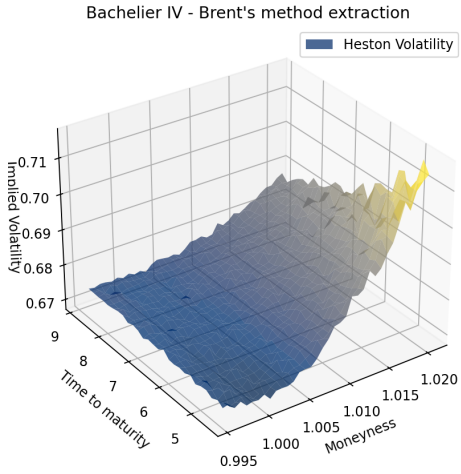
((c)) Bachelier with $\rho = 0.3$, $\kappa = 5$, $\theta = 0.8$, $\nu = 0.3$



((d)) Bachelier with $\rho = 0$, $\kappa = 5$, $\theta = 0.8$, $\nu = 0.3$



((a)) Bachelier with $\rho = 0.3$, $\kappa = 5$, $\theta = 0.46$, $\nu = 0.6$



((b)) Bachelier with $\rho = 0$, $\kappa = 5$, $\theta = 0.46$, $\nu = 0.6$

Figure A.2: Implied Volatility surfaces using Bachelier model with different Heston parameters.

A.4 Generative Artifial Intelligence Disclosure

Generative artificial intelligence (gen AI) tools were not involved in the creation of text, functional pieces of code, or any major part of this project. We only recognize partial assistance of gen AI tools for support in the development of code rendering the graphs, for occasional code debugging, and for Latex document formatting tasks.