Universitat Pompeu Fabra Barcelona

FACULTY OF ECONOMICS AND BUSINESS

End-of-degree Project

Stochastic Volatility Models

Ferran Burgaya Ventura

Tutored by Professor Elisa Alòs

Economics UPF

Academic Year 2020-2021

Code: EMC10

ABSTRACT

The Black Scholes formula is one of the most used to determine what is the fair value of options. However, there are some assumptions such as that volatility is constant that are not met in the real market. Stochastic volatility models can be used as an extension of Black Scholes where volatility is a random process. The aim of this project is to use the SABR model which is a stochastic volatility model to attempt to capture the volatility smile. We study classical problems related to the SABR model such as simulation. Finally, we study its implementation by the financial industry and its main limitations.

Keywords: SABR model, volatility smile, volatility skew, stochastic volatility, volatility surface.

Acknowledgements

This project would not have been possible without the support of my professors, family, and friends. In particular, I would like to extend my sincere thanks to Prof. Elisa Alòs for her guidance at each stage of the project.

Contents

1. Introduction

Black Scholes assumes that volatility is constant. However, this assumption is not met in the real market. Accordingly, we use the Stochastic Alpha, Beta, Rho(α, β, ρ) model (SABR), which assumes that the volatility is a random process. In this project, we do a bibliographic review, where we study some of the classical problems regarding this model, and finally, we study its implementation by the financial industry and its main limitations. We use Python for the plots, tables, simulations, and results. For all calculations, we use my personal laptop. The Processor is Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz. The code is available in the appendix.

This project is comprised of four sections. In section one, we briefly introduce the Black Scholes model and some relevant concepts such as implied volatility. Furthermore, we explain the reason for using stochastic volatility models, as well as introduce the Wiener Process and Itô Process.

After introducing the fundamental concepts of the project, in section two we explain the main features of the SABR model. Additionally, we discuss how the different model parameters behave and their effects on the volatility smile and volatility skew.

In the third section, we study one of the classical problems of the SABR model, which is simulation, for which we introduce the Monte Carlo method. And to reduce the variance of the resulting expectation, we use Conditional Monte Carlo methods.

Finally, in section four we conduct an analysis of the implied volatility through simulation and comparison. And at the end of the project, we draw conclusions according to our results.

2. From constant to Stochastic Volatility

2.1. Black-Scholes

The Black-Scholes model introduced by (Black & Scholes, 1973), is used to calculate option prices. Through a partial differential equation, the price of European call options can be calculated:

$$c = S_0 \mathrm{N}(d_1) - Ke^{-rT}N(d_2)$$
$$d_1 = \frac{\ln S_0 - lnK + rT}{\sigma\sqrt{T}}, \qquad d_2 = \frac{\ln S_0 - lnK + rT}{\sigma\sqrt{T}} - \frac{\sigma\sqrt{T}}{2} \tag{2.1}$$
$$p = Ke^{-rT}N(-d_2) - S_0 \mathrm{N}(-d_1)$$

This model is quite simple, but some non-observable parameters can change option prices. In the real market, volatility σ is not constant. Volatility is a non-observable parameter because it is a random process. For this reason, having a process that depends on random variables over time we use stochastic volatility models.

2.2. Implied Volatility

Implied volatility is a forward-looking measure, it is the option market's estimate of how volatile an option's underlying is going to be during its life (McMillan, L.G, 2019). If we have the option price given by the market, we can invert the Black Scholes formula to obtain the implied volatility.

Expected future volatility plays a very important role in option pricing (Dumas et al., 1998), so estimating this parameter is essential when making financial decisions.

In the real market data, the implied volatility strongly depends on the strike K and time to maturity t_{ex}. To compute the implied volatility, we can solve the BS formula:

$$BS(t_{ex}, S_0, K, r, \sigma_B) = V_0 \tag{2.2}$$

where σ_B denotes the implied volatility and V_0 is the price of options observed in the market.

Volatility is a random process as we will see in the following section, therefore we have to work with models that can reproduce the observed behavior of the implied volatility surface. We do not have an analytical formula to compute σ_B from option prices, hence we need to use numerical methods.

2.3. Stochastic Volatility Models

As we have seen in the previous section, volatility plays an important role in option pricing. The Black-Scholes model for pricing European options is the most widely used formula, also when some of its assumptions are violated, specifically the assumption of having constant volatility.

Assuming the volatility is not a constant, but rather a stochastic process, it is possible to avoid some errors in pricing of Black-Scholes (Ghysels et al., 1996). Stochastic volatility models can be used as an extension of the Black Scholes model where volatility is a random process.

Through implied volatilities for different strike prices, we can create the volatility smile [Figure 1]. Doing the same for different maturities, we obtain the volatility surface [Figure 2], which is a three-dimensional plot of the implied volatilities of an option. Our observations show that for different strike prices there are different implied volatilities, the shape that appears in the first graph is known as the volatility smile. Implicit volatility is greater for options that are deep out-of-the-money or in-the-money, whereas at-the-money options have smaller volatilities (Dumas et al., 1998).

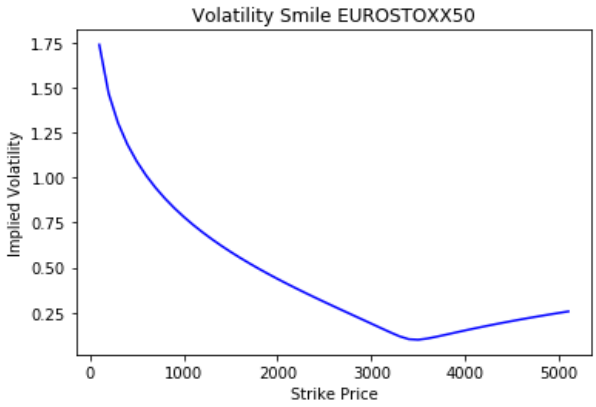


FIGURE 1: Volatility smile. European call options of Eurostoxx50 (March 13, 2019). Data courtesy of Caixa Bank.

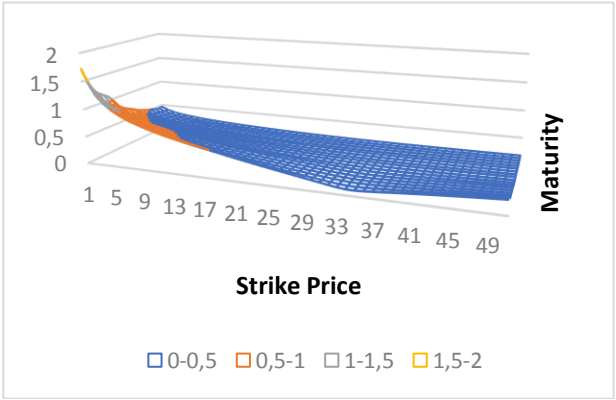


FIGURE 2: Volatility surface. European call options of Eurostoxx50 (March 13, 2019). Data courtesy of Caixa Bank.

2.4. Wiener Process

The Wiener process, also known as Brownian Motion in honor of (Brown, 1828), is a real-valued continuous-time stochastic process $\{W_t\}_{t\geq 0}$. The Brownian W_t is a process such that $W_0 = 0$. The stochastic process $\{W_t\}_{t\geq 0}$ has stationary and independent increments and the increment $W_t - W_s$ has the normal $(0, t-s)$ distribution. W_t has mean zero and the variance is equal to t. The Wiener integral of a function h in the interval $(0,T)$ is denoted by:

$$\int_0^T h(s)dW_s \tag{2.3}$$

and it is a centered Gaussian random variable $N(0, \int_0^T h^2(s)ds)$.

However, in our model we do not have a deterministic h, we have a sigma that expresses stochastic volatility.

2.5. Itô Process

Itô calculus introduced by Kiyoshi Itô can be extended to stochastic processes. Having a random sigma (σ_s) we use the following expression known as the Itô integral:

$$\int_0^T \sigma_s dW_s \ , \tag{2.4}$$

defined as the limit (in an adequate mathematical sense) of Riemann sums.

From this moment on, the expression is not normal because the sigma is random, so the distribution it has is not known.

Using the Itô integral for stochastic volatility models and assuming that r = 0 for simplicity:

$$dS_t = \sigma_t S_t dSW_t \tag{2.5}$$

$$S_T - S_0 = \int_0^T \sigma_t S_t dSW_t \tag{2.6}$$

where the random process would be given by $\sigma_t S_t$.

In stochastic calculus, we use Itô's formula. Given an Itô process of the form $X_t = \int_0^t a_s W_s$ $t \geq 0$, and a function f.

$$df(X_t) = f'(X_t)dX_t + \frac{1}{2}f''(X_t){a_t}^2 dt \tag{2.7}$$

Given a function G, applying Itô's formula to our expression we get:

$$d(G(S_T)) = G'(S_T)\sigma_T S_T dSW_T + \frac{1}{2}G''((S_T)\sigma_T^2 S_T^2 dT \tag{2.9}$$

Some terms are canceled, and we have:

$$d(\ln(S_T)) = \sigma_T dW_t - \frac{1}{2}\sigma_T^2 dT \tag{2.9}$$

Integrating on both sides and doing the same process as before we obtain the following expression:

$$\ln(S_T) - \ln(S_0) = \int_0^T \sigma_T dW_t - \frac{1}{2}\int_0^T \sigma_T^2 dT \tag{2.10}$$

Isolating S_T, we get the final equation:

$$S_T = S_T \exp\left(\int_0^T \sigma_T dW_t - \frac{1}{2}\int_0^T \sigma_T^2 dT\right) \tag{2.11}$$

Having stochastic volatility, prices can be written in this way. In this equation, we have used an interest rate of zero to simplify it, and now only knowing the value of sigma that is a random process we will be able to know which S_T we have.

In this first section we have introduced some of the fundamental concepts of the project, in the following section, we explain the main features of the SABR model.

3. SABR

The SABR model introduced by (Hagan et al., 2002), is a stochastic volatility model that attempts to capture the volatility smile. The model has the parameters: Stochastic Alpha, Beta, Rho(α, β, ρ).

In the SABR model the forward value $\hat{F}$ satisfies:

$$\begin{aligned} d\hat{F} &= \hat{\alpha}\,\hat{F}^{\beta} dpW_t + \sqrt{1-p^2}B, \qquad \hat{F}(0) = f \\ d\hat{\alpha} &= v\hat{\alpha}\,\hat{F}^{\beta} dW_t, \qquad \hat{\alpha}\,(0) = \alpha \end{aligned} \tag{3.1}$$

where the forward value and the volatility $\hat{\alpha}$ are correlated and B is independent of W_t, we can interpret W_t and B as Brownian Motions where $-1 \leq \rho \leq 1$.

Volatility comes from the parameter v or volatility of volatility and $v > 0$. Volatility of volatility is larger for short-term dates, and as the term increases the value of v decreases.

The parameter $\beta \in [0,1]$ has two special cases: $\beta = 0$ represents a normal stochastic model. In contrast, $\beta = 1$ represents a lognormal stochastic model. The choice $\beta = 1$ is the most natural because traders have two parameters that are difficult to differentiate β and rho, for this reason, it is preferred to set β equal to one to differentiate both parameters, as a result, the skew will be based only on rho.

Under the SABR model, we can get the price of European options through the Black Scholes formula. To refer to the volatility of Black Scholes, we use the implied volatility σ_B, through this parameter we can price the options:

$$\begin{aligned} V_{call} &= D(t_{set})\{f\mathrm{N}(d_1) - KN(d_2)\} \\ V_{put} &= V_{call} + D(t_{set})[K - f] \\ d_{1,2} &= \frac{\log\frac{f}{K} \pm \frac{1}{2}\sigma_B^2 t_{ex}}{\sigma_B\sqrt{t_{ex}}} \end{aligned} \tag{3.2}$$

where the implied volatility $\sigma_B(f, K)$ is given by:

$$\begin{aligned} \sigma_B(f,K) = {} & \frac{\alpha}{(fK)^{\frac{1-\beta}{2}}\left\{1 + \frac{(1-\beta)^2}{24}log^2 f/K + \frac{(1-\beta)^4}{1920}log^4 f/K + \cdots\right\}} \cdot \left(\frac{z}{x(z)}\right) \\ & \cdot \left\{1 + \left[\frac{(1-\beta)^2}{24}\frac{\alpha^2}{(fK)^{1-\beta}} + \frac{1}{4}\frac{\rho\beta v\alpha}{(fK)^{(1-\beta)/2}} + \frac{2-3\rho^2}{24}v^2\right]t_{ex}\right\} + \cdots \end{aligned} \tag{3.3}$$

where z and x(z) are the following expressions:

$$z = \frac{v}{\alpha}(f/K)^{(1-\beta)/2} \log f/K$$

$$x(z) = \log\left\{\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}\right\}$$

If we use the special case for the at-the-money options ($K = f$) the volatility formula is reduced:

$$\sigma_{ATM} = \sigma_B(f,f) = \frac{\alpha}{f^{(1-\beta)}}\left\{1+\left[\frac{(1-\beta)^2}{24}\frac{\alpha^2}{f^{2-2\beta}}+\frac{1}{4\frac{\rho\beta\nu\alpha}{f^{(1-\beta)}}}+\frac{2-3\rho^2}{24}v^2\right]t_{ex}\right\} + \cdots \quad (3.4)$$

The "+..." part is omitted because the numbers are very small.

For the lognormal case, when $\beta = 1$ the previous equation reduces to:

$$\sigma_B(f,K) = \alpha\cdot\left(\frac{z}{x(z)}\right)\cdot\left\{1+\left[\frac{1}{4}\rho\nu\alpha+\frac{2-3\rho^2}{24}v^2\right]t_{ex}\right\}+\cdots \quad (3.5)$$

Where:

$$z = \frac{v}{\alpha}log\left(\frac{f}{K}\right)$$

$$x(z) = \log\left\{\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}\right\}$$

In the particular case when the strike $K = f$ we have:

$$\sigma_{ATM} = \sigma_B(f,f) = \alpha\left\{1+\left[\frac{1}{4\rho\nu\alpha}+\frac{2-3\rho^2}{24}v^2\right]t_{ex}\right\}+\cdots \quad (3.6)$$

Through our data, we can make a plot of the different calibrated parameters and see how they change according to maturity. In [Figure 3] we can see how Alpha, Nu, and Rho behave depending on maturity. Alpha seems constant, but it increases slowly over time. Nu decreases over time exponentially; Nu is larger for short-term maturities and decreases for long-term maturities. Rho increases as the maturity increases and it tends to zero.

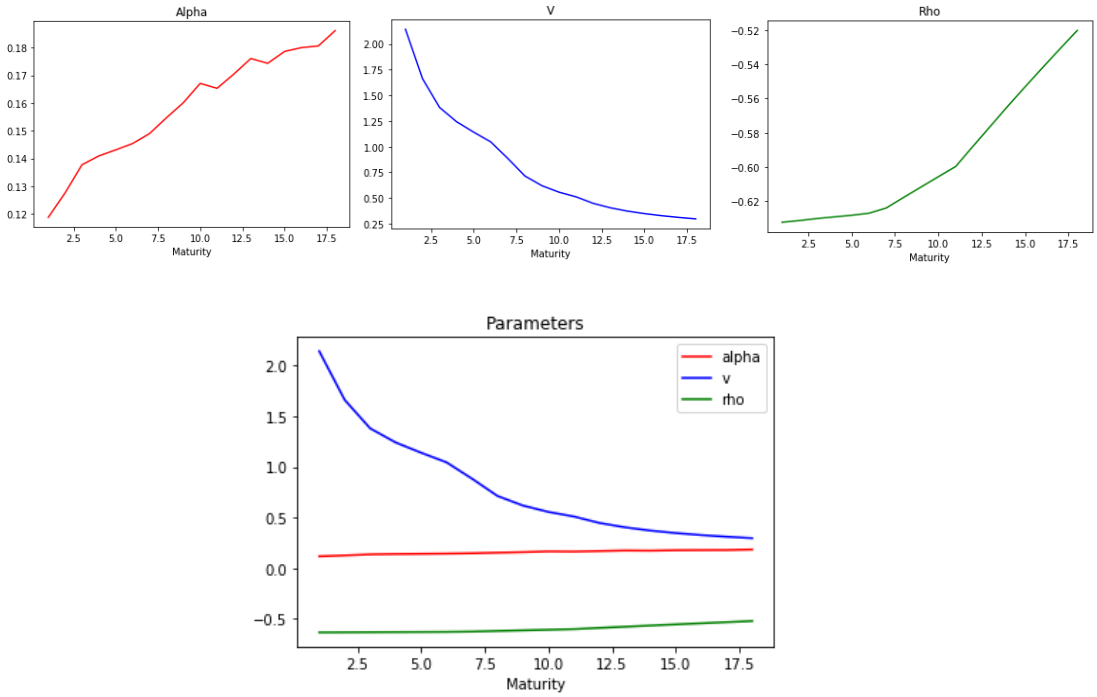


FIGURE 3: Alpha, nu and rho. (March 13, 2019). Data courtesy of Caixa Bank.

We can also add the rho*v parameter as we can see in [Figure 4]:

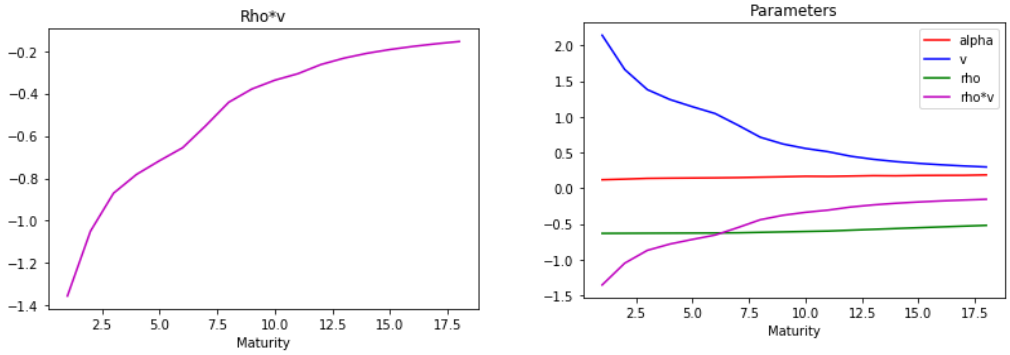


*FIGURE 4: Rho*v. (March 13, 2019). Data courtesy of Caixa Bank.*

Given the complexity of the above formula $\sigma_B(f,K)$, (3.5) can be approximated as:

$$\sigma_B(K,f) = \frac{\alpha}{f^{1-\beta}}\left\{1 - \frac{1}{2}(1-\beta-p\lambda)\log\left(\frac{K}{f}\right) + \frac{1}{12}\left[(1-\beta)^2 + (2-3p^2)\lambda^2\right]\log^2\frac{K}{f}\right\} + \cdots \tag{3.7}$$

where $\lambda = \frac{v}{\alpha} f^{1-\beta}$

Assuming that $\beta = 1$, we can simplify this expression:

$$\sigma_B(K,f) = \alpha\left\{1 - \frac{1}{2}(-p\lambda)\log\left(\frac{K}{f}\right) + \frac{1}{12}\left[(2-3p^2)\lambda^2\right]\log^2\left(\frac{K}{f}\right)\right\} + \cdots \tag{3.8}$$

where $\lambda = \frac{v}{\alpha}$

In [Figure 5] we can see that, when the strike K is not very far from the current forward f, we have good approximations using the simplified formula (3.7). However, when K is very far from f and is deep out-of-the-money or deep in-the-money, the approximations are worse. In these cases, it is better not using the simplified formula.

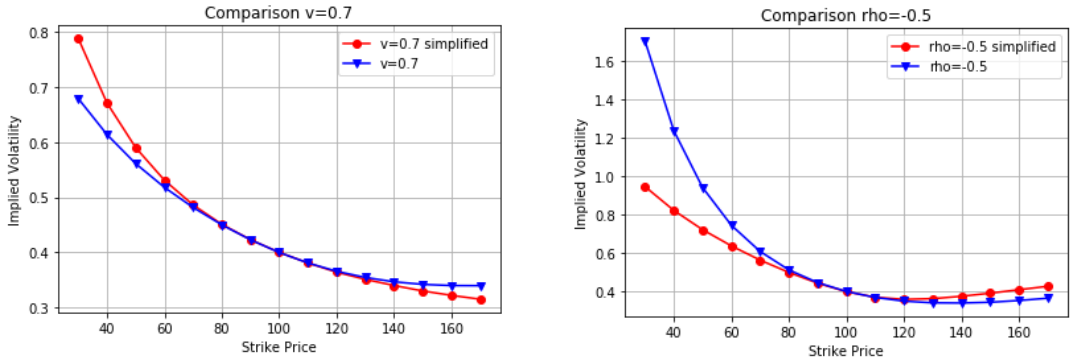


FIGURE 5: Comparison between parameters using the simplified formula (3.7) and not using it (3.5).

3.1. SABR parameters

So far, we have introduced the SABR model that attempts to capture the volatility smile. Now we analyze the impact of the different parameters of the model, using the formula (3.7) with the base parameters: f_0*=100,* v *=1.2,*$\beta = 1$ *rho=-0.6,* α *=0.2, and T=2.*

3.1.1. Effect of T

Through plots of implied volatility for different maturities, we obtain the volatility surface [Figure 2] as we have seen in Section 2.3. The implied volatility is higher for short-dated options and as the time to maturity increases the implied volatility becomes smaller.

3.1.2. Effect of v

In [Figure 6] the effect of v on the smile can be observed. As v increases, the volatility smile is becoming more pronounced (Hagan et al., 2002).

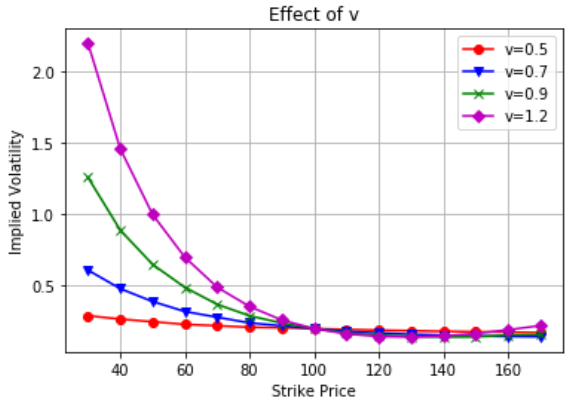


FIGURE 6: Effect of the v parameter on the volatility smile. With the base parameters: f_0*=100,* v *=1.2,*$\beta = 1$ *rho=-0.6,* α *=0.2 and T=2*

3.1.3. Effect of α

In [Figure 7] we can observe that when α decreases the curve moves downwards and when it increases it moves upwards.

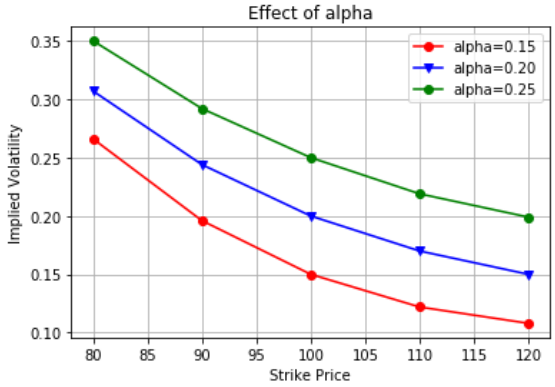
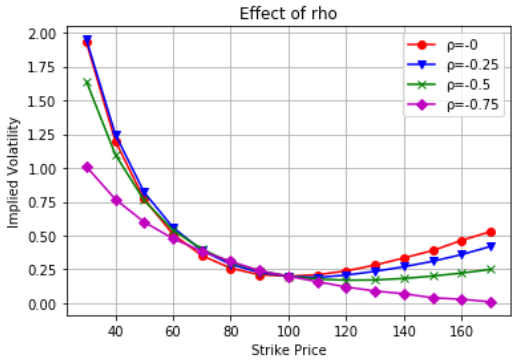


FIGURE 7: Effect of the alpha parameter on the volatility smile. With the base parameters: f_0=100, v =1.2,$\beta = 1$ rho=-0.6, α =0.2 and T=2

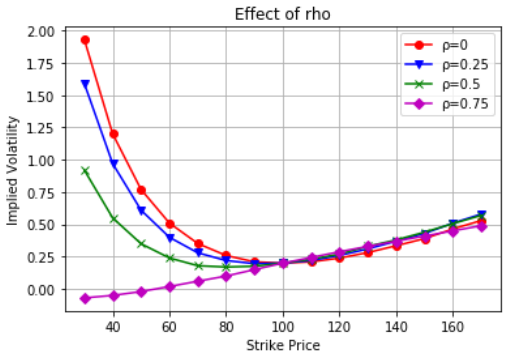
3.1.4. Effect of rho

As can be seen in Section 3, the rho values range from $-1 \leq \rho \leq 1$. To see how they behave, we make different plots for positive and negative values of rho.

In [Figure 8] we can observe the effect of rho on the volatility skew. When rho has negative values, the smaller the rho, the volatility skew increases as we can see in a). The opposite occurs if the value of rho is positive as can be seen in b).



a) Negative values of rho



b) Positive values of rho

FIGURE 8: Effect of the rho parameter on the volatility skew. With the base parameters: f_0=100, v =1.2,$\beta = 1$ rho=-0.6, α =0.2 and T=2

3.1.5. Effect of β

As we have already seen at the beginning of this section, β and rho are very similar and to simplify our expression we use the value $\beta = 1$.

4. Simulations

After introducing the SABR model and how its parameters behave in section 3. In this section, we study one of the classical problems of the SABR model, which is simulation.

4.1. Monte Carlo

The Monte Carlo method is used to approximate complex mathematical formulas through multiple simulations (Ulam, 1949). The error can be estimated through the central limit theorem, and as the sample is larger the error converges to zero (Broadie & Kaya, 2006).

4.2. Conditional Monte Carlo

Monte Carlo methods can sometimes be very slow, for this reason, we use Conditional Monte Carlo methods that condition the expected value and can be used to determine the price of options. Conditional Monte Carlo methods are often used to reduce the variance of the resulting expectation (Hirsa, 2011). Now we have the following expression:

$$v_t = \sqrt{\frac{1}{T-t}\int_t^T \sigma_s^2 ds} \tag{4.1}$$

where v_t represents the future mean volatility.

Knowing that this expression holds:

$$E(S_t - K)_+ = E(BS(T, S_t, K, v_t)) \tag{4.2}$$

Using the concept of conditional expectation, we can write the following expression, that is the conditional expectation when W is deterministic:

$$E\big(BS(T, S_t, K, v_t)\big) = E[E(BS(T, S_t, K, v_t)|W)] \tag{4.3}$$

Initial asset price is given by:

$$S_0' = S_0 \exp\left(-0.5\rho^2 \int_0^T \sigma_s^2 ds + \rho \int_0^T \sigma_s \, (dW_s)\right) \tag{4.4}$$

And S is a process with deterministic volatility given by:

$$\sqrt{(1-\rho^2)}\sigma \tag{4.5}$$

Now we have the conditional expectation changing the initial condition S_0' and the volatility.

Then we get:

$$E(BS(T, S_t, K, v_t)|W) = BS(0, S_0', K, \sqrt{(1-\rho^2)}v_0) \tag{4.6}$$

Which implies that:

$$E(S_t - K)_+ = E(BS(T, S_t, K, v_t)) = BS(0, S_0', K, \sqrt{(1-\rho^2)}v_0) \tag{4.7}$$

Formula (4.7) was developed by (Willard, 1997) and (Romano & Touzi, 1997).

4.3. CMC on SABR

Using conditional expectations (Willard, 1997) and (Romano & Touzi, 1997), we can compute the Monte Carlo prices following the next steps:

i. Simulate the processes W and σ
ii. Compute S_0'
iii. Compute v_0
iv. Compute $BS(0, S_0', K, \sqrt{(1-\rho^2)}v_0)$
v. Repeat step iv. n times to get n simulations of $BS(0, S_0', K, \sqrt{(1-\rho^2)}v_0)$ and take the mean.

In [Figure 9] we can observe two of the simulations of the Brownian motion W and σ. Using the Monte Carlo Method described above, many simulations are performed and through them, we can calculate the future expected value of σ for different maturities.

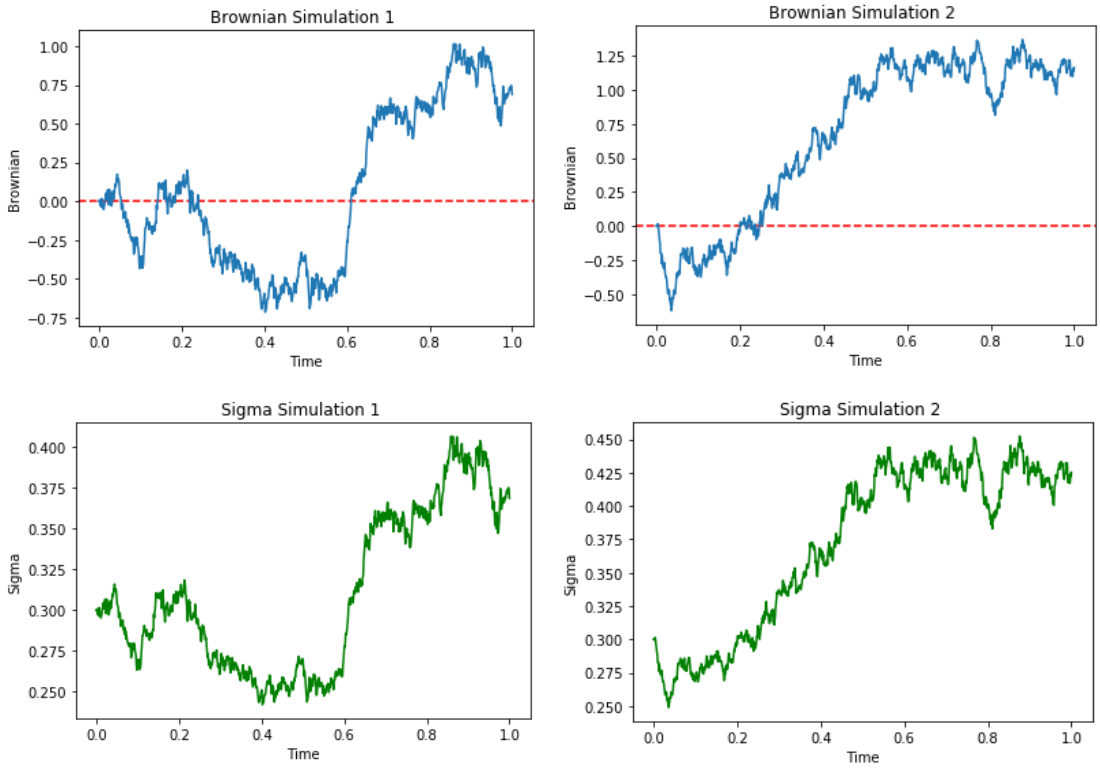


FIGURE 9: Monte Carlo Simulation paths of the Brownian motion W and sigma with the base parameters $T = 1$ and $\sigma = 0.3$.

After having seen how the first step is carried out, we will see two different cases of the model, when |rho|=0 and when |rho|>0.

4.3.1. |rho|=0

When |rho|=0, we can observe that some of the model's formulas are simplified. Formula (4.4) reduces to $S_0' = S_0$ and formula (4.5) reduces to σ. This implies that volatility and prices are independent.

In [Table 1] the numerical results of performing the different steps described above can be observed. [Table 1] describes the runtime per simulation in seconds, the mean of the results after fifty simulations, and the standard deviation.

Method	Time Steps m	No. of simulation trials	Time (Seconds)	Mean	stdev
CMC	1000	100	2.89	12.9123	0.4006
CMC	1000	5000	49.88	12.7906	0.0578
CMC	1000	10000	113.07	12.8777	0.039

TABLE 1: Numerical results when |rho|=0. Base parameters: $K = S_0 = 100, r = 0, nu = 0.5, \sigma = 0.3$ *and* $T = 1$.

In [Figure 10] we can observe the standard deviation and the time as the number of simulation trials increases. The larger the n, the longer the time per simulation. However, we obtain better results since when n increases, the standard deviation is smaller.

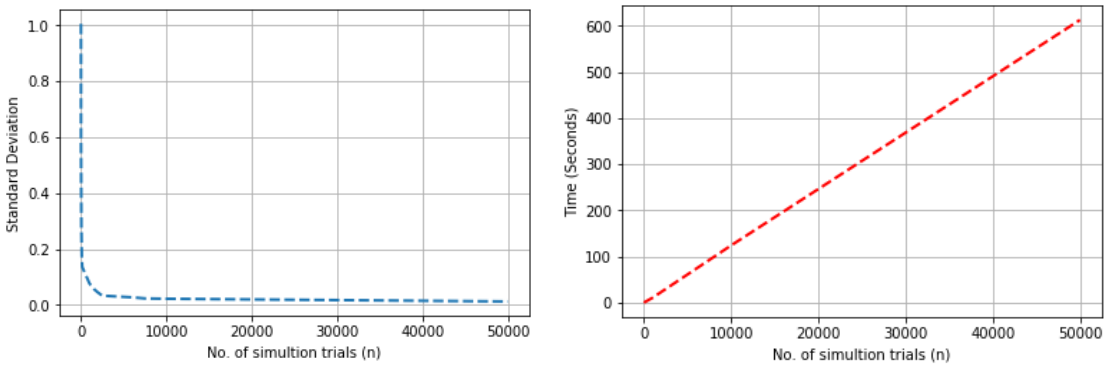


FIGURE 10: Standard deviation and time per simulation as n increases.

In [Table 2] we can see that by increasing T the mean value increases. And as in [Table 1], the larger the n, the smaller the standard deviation.

Method	Time Steps m	No. of simulation trials	Time (Seconds)	Mean	stdev
CMC	1000	100	2.53	19.6764	1.0425
CMC	1000	5000	49.88	19.3302	0.14194
CMC	1000	10000	113.29	19.4753	0.0899

TABLE 2: Numerical results when |rho|=0. Base parameters: $K = S_0 = 100, r = 0, nu = 0.5, \sigma = 0.3$ *and* $T = 2$.

4.3.2. |rho|>0

Now we follow the same steps as before, however, since |rho|>0, we change the condition S_0' as we can see in formula (4.4) and we also change the volatility as we can see in formula (4.5). The higher the values of |rho|, the higher the correlation with σ.

In [Table 3] and [Table 4] the numerical results when |rho| is greater than zero can be observed. The standard deviation, as in the previous tables, decreases when the number of simulation trials is larger.

Method	Time Steps m	No. of simulation trials	Time (Seconds)	Mean	stdev
CMC	1000	100	1.44	12.543	0.5786
CMC	1000	5000	94.42	12.4285	0.0682
CMC	1000	10000	191.74	12.5049	0.0623

TABLE 3: Numerical results when |rho|>0. Base parameters: $K = S_0 = 100, r = 0, nu = 0.5, rho = -0.5, \sigma = 0.3$ *and* $T = 1$.

Method	Time Steps m	No. of simulation trials	Time (Seconds)	Mean	stdev
CMC	1000	100	1.29	18.2299	0.7062
CMC	1000	5000	97.97	18.1323	0.1145
CMC	1000	10000	190.54	18.3343	0.074

TABLE 4: Numerical results when |rho|>0. Base parameters: $K = S_0 = 100,\ r = 0,\ nu = 0.5,\ rho = -0.5,\ \sigma = 0.3$ *and* $T = 2$.

5. Results

After having seen a classical problem of the SABR model such as simulation, before reaching the conclusions we analyze implied volatility by simulation and comparison.

5.1. Analysis of implied volatility

Through formulas defined in section 4, the Conditional Monte Carlo simulation's function is optimized. To do this we have calculated a function in order to minimize the error using Python (See appendix 5). Through this function, the implied volatilities are calculated for different strikes with the same time to maturity T.

In [Figure 11] we can observe the result of previous calculations with $rho = 0$, which create the volatility smile. For ATM options the implied volatility is smaller and for out-of-the-money and in-the-money calls the implied volatility is higher. In [Figure 12] we can observe the volatility skew when $rho \neq 0$.

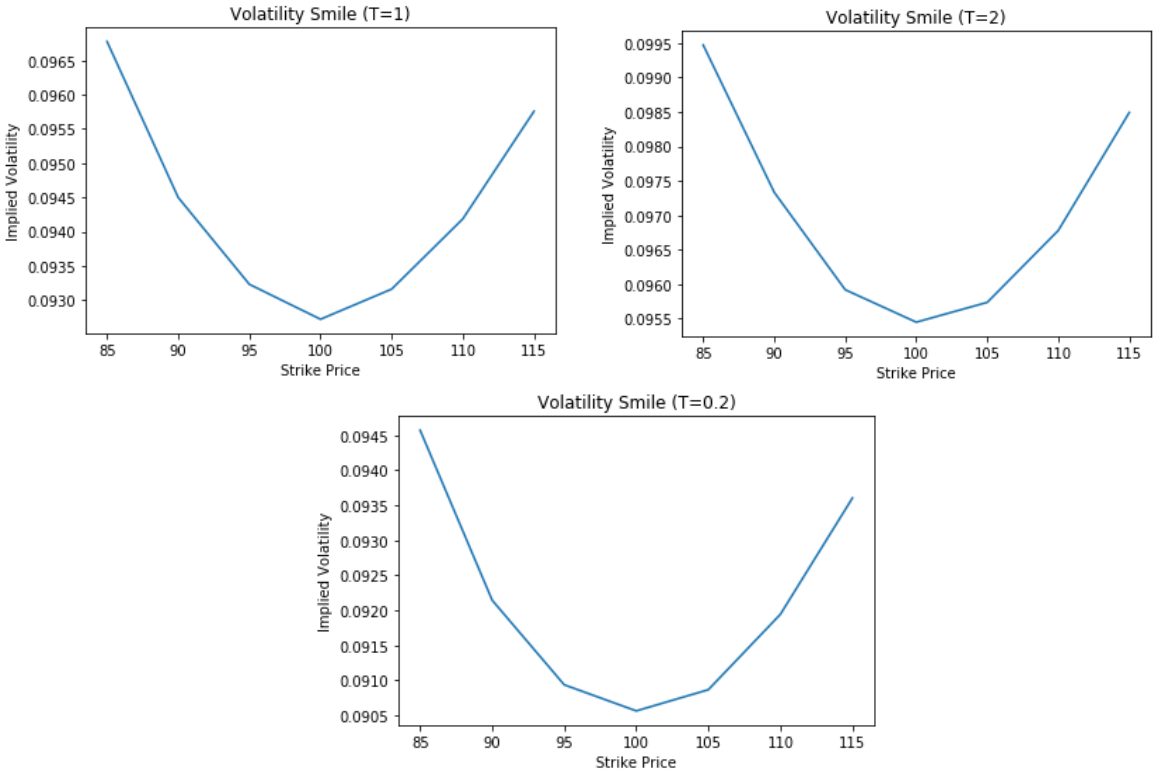


FIGURE 11: Implied volatility using the CMC method for different strikes. With the base parameters: r=0, S_0=100, nu=0.3, alpha=0.09, rho=0, n=100000

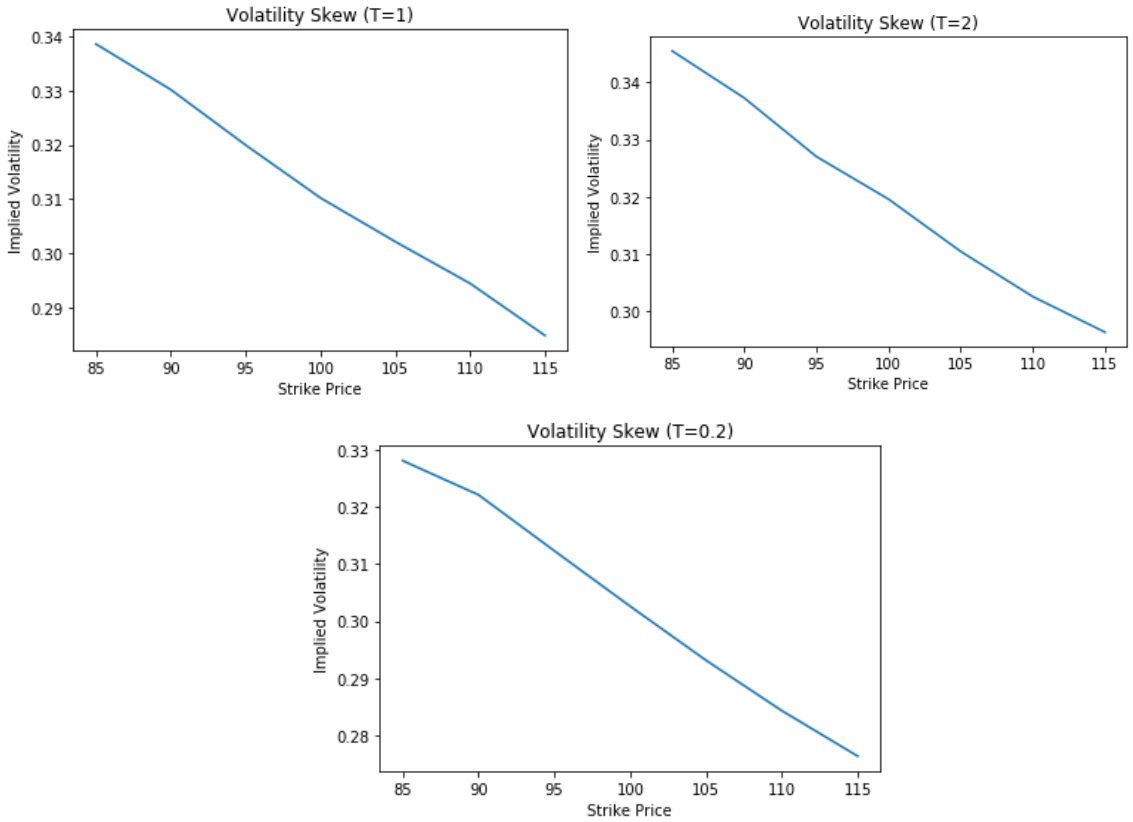


FIGURE 12: Implied volatility using the CMC for different strikes. With the base parameters: r=0, S_0=100, nu=0.5, alpha=0.3, rho=-0.75, n=100000

6. Conclusion

This research aimed to understand the use of stochastic volatility models, specially the SABR model, and identify its main limitations. To sum up:

- Stochastic volatility models can produce more realistic estimates because it incorporates the assumption that volatility is a random process.
- Implied volatility is higher for short-dated options and becomes smaller as the time to maturity increases.
- As v increases, the volatility smile becomes more pronounced.
- The curve moves in the same direction as α. When α decreases the curve moves downwards and when it increases it moves upwards.
- For negative values of rho, the smaller the rho, the volatility skew increases. The opposite occurs for positive values of rho.
- Conditional Monte Carlo method can be quite effective in calculating the price of options. However, this method takes a long time to do the calculations.
    - Variance is greatly reduced by increasing the number of simulations. However, time per simulation increases.
- Hagan's formula takes much less time compared to the CMC method.

For future extensions, other models attempt to solve the problem of the Black-Scholes constant volatility assumption such as the Heston model (Heston, 1993), the CEV model (Cox, 1975), the GARCH model (Bollerslev, 1986), or some extensions of the SABR model for negative rates (Antonov et al., 2015). Other approaches include the inclusion of jump processes (Cont and Tankov, 2004) or models based on other Gaussian processes different from the Brownian motion (see Comte and Renault, 1998 and Alòs León Vives, 2007, among others). Therefore, in the future, more research should be done.

References

Alòs, E. (2019). Advanced Option Pricing and Modeling: Stochastic volatility models, 1-36.

Alòs, E., León, J. A., & Vives, J. (2007). On the short-time behavior of the implied volatility for jump-diffusion models with stochastic volatility. *Finance and Stochastics*, *11*(4), 571-589.

Antonov, A., Konikov, M., & Spector, M. (2015). The free boundary SABR: natural extension to negative rates. Available at SSRN 2557046.

Black, & Scholes, M. (1973). Black Scholes - 1972 - Pricing of options.pdf. In *The Journal of Political Economy* (Vol. 81, Issue 3, pp. 637–654).

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of econometrics, 31(3), 307-327.

Broadie, M., & Kaya, Ö. (2006). Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations Research*, *54*(2), 217–231. https://doi.org/10.1287/opre.1050.0247

Brown, R. (1828). XXVII. A brief account of microscopical observations made in the months of June, July and August 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies . *The Philosophical Magazine*, *4*(21), 161–173. https://doi.org/10.1080/14786442808674769

Comte, F., & Renault, E. (1998). Long memory in continuous-time stochastic volatility models. *Mathematical finance*, *8*(4), 291-323.

Cont, R., & Tankov, P. (2004). Nonparametric calibration of jump-diffusion option pricing models. *The Journal of Computational Finance*, *7*, 1-49.

Cox, J. (1975). Notes on option pricing I: Constant elasticity of variance diffusions. Unpublished note, Stanford University, Graduate School of Business.

Deloitte. (2016). *Risk management under the SABR model*. *October*, 0–17.

Dumas, B., Fleming, J., & Whaley, R. E. (1998). Implied volatility functions: Empirical tests. In *Journal of Finance* (Vol. 53, Issue 6, pp. 2059–2106). https://doi.org/10.1111/0022-1082.00083

Ghysels, E., Harvey, A. C., & Renault, E. (1996). 5 Stochastic volatility. *Handbook of Statistics*, *14*(December), 119–191. https://doi.org/10.1016/S0169-7161(96)14007-4

Hagan, P. S., Kumar, D., Lesniewski, A. S., & Woodward, D. E. (2002). Managing smile risk. *Wilmott Magazine*, *m*, 84–108. http://www.math.columbia.edu/~lrb/sabrAll.pdf

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. The review of financial studies, 6(2), 327-343.

Hirsa, A. (2011). Computational Methods in Finance CRC, Boca Raton, FL; 244-245

Mcmillan, L.G. (2012). Options as a Strategic Investment. Penguin; 862-865

Romano, M., & Touzi, N. (1997). Contingent claims and market completeness in a stochastic volatility model. *Mathematical Finance*, *7*(4), 399–412. https://doi.org/10.1111/1467-9965.00038

Ulam, N. M. S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, *44*(247), 335–341.

Willard, G. (1997). Calculating prices and sensitivities for path-independent derivative securities in multifactor models. *Journal of Derivatives, vol. 5*, pp. 45-61

Appendix

Python code

Appendix 1. Sigma simulations

```
import numpy as np
import pandas as pd
import scipy.stats as ss
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import math
from tabulate import tabulate
import timeit

#Black Scholes

def d1(S0,K,r,sigma,T):
    d1 = (np.log(S0/K)+(r+sigma**2/2)*T)/(sigma*np.sqrt(T))
    return d1

def d2(S0,K,r,sigma,T):
    d2 = (np.log(S0/K)+(r-sigma**2/2)*T)/(sigma*np.sqrt(T))
    return d2

def BlackScholes(S0,K,r,sigma,T):
    blacks        =        S0*ss.norm.cdf(d1(S0,K,r,sigma,T))-K*np.exp(-
r*T)*ss.norm.cdf(d2(S0,K,r,sigma,T))
    return blacks

def VCall(S0,K,r,sigma,Tex):
    call        =        S0*ss.norm.cdf(d1(S0,K,r,sigma,Tex))-K*np.exp(-
r*Tex)*ss.norm.cdf(d2(S0,K,r,sigma,Tex))
    return call

def VPut(S0,K,r,sigma,Tex):
    put      =      K*np.exp(-r*Tex)*ss.norm.cdf(-d2(S0,K,r,sigma,Tex))-
S0*ss.norm.cdf(-d1(S0,K,r,sigma,Tex))
    return put

#Sigma simulations

T=1
r=0
n=10
sigma0=0.2

brownian=np.zeros([n,1001])
sigma=np.zeros([n,1001])
time=np.zeros([1001])
W=np.random.normal(0,1,(n,1001))

for j in range (0,n):

    sigma[j,0]=sigma0
    brownian[j,0]=0
```

```
    for i in range (0,1000):
        brownian[j,i+1]=brownian[j,i]+W[j,i]*np.sqrt(T/1000)
        time[i+1]=time[i]+T/1000
        sigma[j,i+1]=sigma[j,0]*np.exp(((r-
(0.5)*sigma0**2))*time[j+1]+sigma0*brownian[j,i+1])

    plt.plot(time, sigma[j])
plt.show()
```

Appendix 2. Volatility Smile and Volatility Surface

```
#Volatility smile and volatility surface

df = pd.read_excel('C:/Users/ferran/Desktop/DataTFG/data.xlsx')
df2 = pd.read_excel('C:/Users/ferran/Desktop/DataTFG/data2.xlsx')

#Volatily smile

plt.title('Volatility Smile EUROSTOXX50')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df2.strike,df2.IV,"r")
plt.grid()
plt.show()

#Parameters

plt.title('Rho')
plt.xlabel('Maturity')
plt.ylabel('')
plt.plot(df2.maturity,df2.rho,"g")
plt.grid()
plt.show()

plt.title('V')
plt.xlabel('Maturity')
plt.ylabel('')
plt.plot(df2.maturity,df2.v,"b")
plt.grid()
plt.show()

plt.title('Alpha')
plt.xlabel('Maturity')
plt.ylabel('')
plt.plot(df2.maturity,df2.alpha,"r")
plt.grid()
plt.show()

plt.title('Rho*v')
plt.xlabel('Maturity')
plt.ylabel('')
plt.plot(df2.maturity,df2.rhov,"m")
plt.grid()
plt.show()
```

```
plt.plot(df2.maturity,df2.alpha,"r")
plt.plot(df2.maturity,df2.v,"b")
plt.plot(df2.maturity,df2.rho,"g")
#plt.plot(df2.maturity,df2.rhov,"m")
plt.title('Parameters')
plt.xlabel('Maturity')
plt.legend(['alpha','v','rho','rho*v'])
plt.grid()
plt.show()
```

Appendix 3. Parameters

```
df = pd.read_excel('C:/Users/ferran/Desktop/DataTFG/datap.xlsx')

#Hagan Formula

def IVHagan(S0,K,nu,sigma0,rho,T,beta):

    if abs(S0-K)<0.0001: #ATM Volatility
        var1 = sigma0/S0**(1-beta)
        var2             =             (1-beta)**2/24*(sigma0**2/S0**(2-
2*beta))+1/4*rho*beta*nu*sigma0/S0**(1-beta) + (2-3*rho**2)*nu**2/24
        IV = var1*(1+var2*T)

    else:               #Non-ATM Volatility
        z=nu/sigma0*np.log(S0/K)
        x=np.log((np.sqrt(1-2*rho*z+z**2)+z-rho)/(1-rho))
        IV= sigma0*(z/x)*(1+(1/4*(rho*nu*sigma0)+((2-3*rho**2)/24))*T)

    return IV

#Simplified Hagan Formula

def SHagan(f,K,alpha,rho,v):

    lam=v/alpha

    sIV=alpha*(1-1/2*(-rho*lam)*np.log(K/f)+1/12*((2-
3*rho**2)*lam**2*np.log(K/f)**2))

    return sIV


#Effect of SABR parameters on the smile with base parameters: f=100,
v=1.5, rho=-0.6, alpha=0.4, and T=2

plt.title('rho effect (T=0.1)')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.I,"r-o")
plt.plot(df.K,df.II,"b-v")
plt.plot(df.K,df.III,"g-x")
plt.plot(df.K,df.IV,"m-D")
plt.legend(['rho=-0.9','rho=-0.5','rho=-0.25','rho=-0'])
plt.grid()
plt.show()
```

```
plt.title('rho effect (T=2)')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.V,"r-o")
plt.plot(df.K,df.VI,"b-v")
plt.plot(df.K,df.VII,"g-x")
plt.plot(df.K,df.VIII,"m-D")
plt.legend(['rho=-0.9','rho=-0.5','rho=-0.25','rho=-0'])

plt.grid()
plt.show()

plt.title('v effect (T=0.1)')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.IX,"r-o")
plt.plot(df.K,df.X,"b-v")
plt.plot(df.K,df.XI,"g-x")
plt.plot(df.K,df.XII,"m-D")
plt.legend(['v=0.5','v=0.7','v=0.9','v=1.2'])

plt.grid()
plt.show()

plt.title('v effect (T=2)')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.XIII,"r-o")
plt.plot(df.K,df.XIV,"b-v")
plt.plot(df.K,df.XV,"g-x")
plt.plot(df.K,df.XVI,"m-D")
plt.legend(['v=0.5','v=0.7','v=0.9','v=1.2'])

plt.grid()
plt.show()

#Using the simplified Hagan formula

plt.title('p effect simplified formula')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.XVII,"r-o")
plt.plot(df.K,df.XVIII,"b-v")
plt.plot(df.K,df.XIX,"g-x")
plt.plot(df.K,df.XX,"m-D")
plt.legend(['rho=-0.9','rho=-0.5','rho=-0.25','rho=-0'])

plt.grid()
plt.show()

plt.title('v effect simplified formula')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.XXI,"r-o")
plt.plot(df.K,df.XXII,"b-v")
plt.plot(df.K,df.XXIII,"g-x")
plt.plot(df.K,df.XXIV,"m-D")
```

```
plt.legend(['v=0.5','v=0.7','v=0.9','v=1.2'])

plt.grid()
plt.show()

#Comparisons between simplified and not simplified Hagan formula

plt.title('Comparison v=0.5')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.XXI,"r-o")
plt.plot(df.K,df.IX,"b-v")

plt.legend(['v=0.5 simplified','v=0.5'])
plt.grid()
plt.show()

plt.title('Comparison v=0.7')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.XXII,"r-o")
plt.plot(df.K,df.X,"b-v")

plt.legend(['v=0.7 simplified','v=0.7',])
plt.grid()
plt.show()

plt.title('Comparison rho=-0.9')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.I,"r-o")
plt.plot(df.K,df.XVII,"b-v")

plt.legend(['rho=-0.9 simplified','rho=-0.9',])
plt.grid()
plt.show()

plt.title('Comparison rho=-0.5')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.K,df.II,"r-o")
plt.plot(df.K,df.XVIII,"b-v")

plt.legend(['rho=-0.5 simplified','rho=-0.5',])
plt.grid()
plt.show()

#Effect of SABR parameters on the smile with base parameters: f=100,
v=1.2, Beta=1, rho=-0.6, alpha=0.2, and T=1

#Effect of alpha

plt.title('Effect of alpha')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(df.alpha,df.a,"r-o")
plt.plot(df.alpha,df.aa,"b-v")
```

```
plt.plot(df.alpha,df.aaa,"g-o")

plt.legend(['alpha=0.15','alpha=0.20','alpha=0.25'])
plt.grid()
plt.show()

#Effect of v

plt.title('Effect of v')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')

plt.plot(df.K,df.v,"r-o")
plt.plot(df.K,df.vv,"b-v")
plt.plot(df.K,df.vvv,"g-x")
plt.plot(df.K,df.vvvv,"m-D")
plt.legend(['v=0.5','v=0.7','v=0.9','v=1.2'])

plt.grid()
plt.show()

#Effect of rho

plt.title('Effect of rho')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')

plt.plot(df.K,df.p,"r-o")
plt.plot(df.K,df.pp,"b-v")
plt.plot(df.K,df.ppp,"g-x")
plt.plot(df.K,df.pppp,"m-D")
plt.legend(['ρ=-0','ρ=-0.25','ρ=-0.5','ρ=-0.75'])

plt.grid()
plt.show()

plt.title('Effect of rho')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')

plt.plot(df.K,df.P,"r-o")
plt.plot(df.K,df.PP,"b-v")
plt.plot(df.K,df.PPP,"g-x")
plt.plot(df.K,df.PPPP,"m-D")
plt.legend(['ρ=0','ρ=0.25','ρ=0.5','ρ=0.75'])

plt.grid()
plt.show()
```

Appendix 4. Simulations

```
#Brownian and sigma plots

T=1
r=0
n=10
```

```
sigma0=0.3

brownian=np.zeros([n,1001])
sigma=np.zeros([n,1001])
time=np.zeros([1001])
W=np.random.normal(0,1,(n,1001))

for j in range (0,n):

    sigma[j,0]=sigma0
    brownian[j,0]=0

    for i in range (0,1000):
        brownian[j,i+1]=brownian[j,i]+W[j,i]*np.sqrt(T/1000)
        time[i+1]=time[i]+T/1000
        sigma[j,i+1]=sigma[j,0]*np.exp(((r-
(0.5)*sigma0**2))*time[j+1]+sigma0*brownian[j,i+1])

plt.title('Brownian Simulation 2 ')
plt.xlabel('Time')
plt.axhline(y=0,c='r',ls='--')
plt.ylabel('Brownian')
plt.plot(time,brownian[1])

plt.show()
plt.title('Sigma Simulation 2 ')
plt.xlabel('Time')
plt.ylabel('Sigma')
plt.plot(time,sigma[1],"g")
plt.show()

#CMC when |rho|=0

def CMC(T,r,S0,K,nu,sigma0,n):
    time=np.zeros([100001])
    W=np.random.normal(0,1,(n,1001))
    brownian=np.zeros([n,1001])
    sigma=np.zeros([n,1001])
    BS2=np.zeros(n)
    time=np.zeros(100001)
    BSPrice=0
    BS2=np.zeros(n)
    sigma2=np.zeros(n)
    for j in range (0,n):
        sigma[j,0]=sigma0
        brownian[j,0]=0
        for i in range (0,1000):
            brownian[j,i+1]=brownian[j,i]+W[j,i]*np.sqrt(T/1000)
            time[i+1]=time[i]+T/1000
            sigma[j,i+1]=sigma[j,0]*np.exp(((r-
(0.5)*nu**2))*time[j+1]+nu*brownian[j,i+1])
        sigma2[j]=np.sqrt(np.mean(sigma[j]**2))
        BS2[j]=BlackScholes(S0,K,r,sigma2[j],T)
    BSPrice=np.mean(BS2)
    return BSPrice
```

```
#Simulations (n=50)

start = timeit.default_timer()
simulacio=np.zeros([50])
for i in range(0,50):
    simulacio[i]=CMC(1,0,100,100,0.5,0.3,100)
stop = timeit.default_timer()
time= stop - start

print(np.mean(simulacio[i]),np.std(simulacio),time/50)

#mean= mean of results with n=50
#time= time per simulation
#base parameters: nu=0.5, sigma=0.3, rho=0, S0=K=100, T=1

table = [["Method","Time Steps m","No. of simulation trials","Time
(Seconds)","Mean","stdev"],
         ["CMC","1000","100",2.89,12.9123,0.4006],
         ["CMC","1000","5000",49.88,12.7906,0.0578],
         ["CMC","1000","10000",113.07,12.8777,0.039]]

print(tabulate(table, tablefmt='fancy_grid'))

#mean= mean of results with n=50
#time= time per simulation
#base parameters: nu=0.5, sigma=0.3, rho=0, S0=K=100, T=2


table2 = [["Method","Time Steps m","No. of simulation trials","Time
(Seconds)","Mean","stdev"],
         ["CMC","1000","100",2.53,19.6764,1.0425],
         ["CMC","1000","5000",49.88,19.3302,0.14194],
         ["CMC","1000","10000",113.29,19.4753,0.0899]]

print(tabulate(table2, tablefmt='fancy_grid'))

#stdev plot

stdev=[1.005,0.5889,0.2845,0.1762,0.1359,0.0769,0.0573,0.04423,0.0331,
0.0271,0.0232,0.0221,0.0124]
n=[5,10,50,100,150,1000,1500,2000,2500,6000,7500,10000,50000]
plt.xlabel("No. of simultion trials (n)")
plt.ylabel("Standard Deviation")
plt.grid()
plt.plot(n,stdev,lw=2,ls='--')
plt.show()

#Runtime per simulation in seconds

n=[0,10,100,1000,10000,50000]
time=[0,0.12,1.089,10.573,123.9,612.54]
plt.xlabel("No. of simultion trials (n)")
plt.ylabel("Time (Seconds)")
plt.grid()
plt.plot(n,time,"r",lw=2,ls='--')
plt.show()
```

```
#CMC with rho!=0

def CMC2(T,r,S0,K,nu,sigma0,rho,n):
    time=np.zeros([100001])
    W=np.random.normal(0,1,(n,1001))
    brownian=np.zeros([n,1001])
    sigma=np.zeros([n,1001])
    BS2=np.zeros(n)
    S02=np.zeros(n)
    time=np.zeros(100001)
    sigma2=np.zeros(n)
    sigma3=np.zeros(n)
    var2=np.zeros(n)
    for j in range (0,n):
        sigma[j,0]=sigma0
        brownian[j,0]=0
        for i in range (0,1000):
            brownian[j,i+1]=brownian[j,i]+W[j,i]*np.sqrt(T/1000)
            time[i+1]=time[i]+T/1000
            sigma[j,i+1]=sigma[j,0]*np.exp((-
0.5*nu**2)*time[j+1]+nu*brownian[j,i+1])
        sigma2[j]=np.sqrt(np.mean(sigma[j]**2))
        var=np.sum(sigma[j]**2)*T/1000
        for i in range(0,1000):
            var2[j]=var2[j]+sigma[j,i]*(brownian[j,i+1]-brownian[j,i])
        S02[j]=S0*np.exp(-0.5*rho**2*var+rho*var2[j])
        sigma3[j]=np.sqrt(1-rho**2)*sigma2[j]
        BS2[j]=BlackScholes(S02[j],K,r,sigma3[j],T)
    BSPrice=np.mean(BS2)
    return BSPrice

#Simulations s=100, n=50

start = timeit.default_timer()
simulacio=np.zeros([50])
for i in range(0,50):
    simulacio[i]=CMC2(1,0,100,100,0.5,0.3,-0.5,100)
m1=np.mean(simulacio)
s1=np.std(simulacio)
stop = timeit.default_timer()
t1= stop - start

print('mean',m1,'stdev',s1,'Time',t1/50)

#mean= mean of results with n=50
#time= time per simulation
#base parameters: sigma=0.3, rho=-0.5, S0=K=100, T=1

table2 = [["Method","Time Steps m","No. of simulation trials","Time
(Seconds)","Mean","stdev"],
          ["CMC","1000","100",1.44, 12.543,0.5786],
          ["CMC","1000","5000",94.42,12.4285,0.0682],
          ["CMC","1000","10000",191.74,12.5049,0.0623]]

print(tabulate(table2, tablefmt='fancy_grid'))

#mean= mean of results with n=50
```

```
#time= time per simulation
#base parameters: sigma=0.3, rho=-0.5, S0=K=100, T=2


table2 = [["Method","Time Steps m","No. of simulation trials","Time
(Seconds)","Mean","stdev"],
          ["CMC","1000","100",1.29,18.2299,0.7062],
          ["CMC","1000","5000",97.97,18.1323,0.1145],
          ["CMC","1000","10000",190.54,18.3343,0.074]]

print(tabulate(table2, tablefmt='fancy_grid'))
```

Appendix 5. Analysis of implied volatility

```
from scipy import optimize
from scipy.optimize import minimize

#rho=0

def impliedvol(T,K):
    optionprice=CMC2(T,0,100,K,0.3,0.09,0,100000)
    def error(a):
        return (BlackScholes(100,K,0,a,T)-optionprice)**2
    return(optimize.brent(error,brack=(0.0001,5)))

strike=np.zeros(7)
implied=np.zeros(7)
for j in range(0,7):
    strike[j]=85+5*j
    implied[j]=impliedvol(1,strike[j])
    print(strike[j],implied[j])

plt.title('Volatility Smile (T=1)')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(strike,implied)

#rho!=0

def impliedvol(T,K):
    optionprice=CMC2(T,0,100,K,0.09,0.3,-0.75,100000)
    def error(a):
        return (BlackScholes(100,K,0,a,T)-optionprice)**2
    return(optimize.brent(error,brack=(0.0001,5)))

strike=np.zeros(7)
implied=np.zeros(7)
for j in range(0,7):
    strike[j]=85+5*j
    implied[j]=impliedvol(1,strike[j])
    print(strike[j],implied[j])

plt.title('Volatility Skew (T=1)')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.plot(strike,implied)
```