

# Calibrating a SABR Model on Market Data

---

Mario Damiano Russo

Jan 2020

# Table of Contents

An overview of the SABR Model

The Dataset

Calibrating the Model

Preprocessing

Code Implementation

Choosing  $\beta$

Results Showcase

## An overview of the SABR Model

---

# The Black-Scholes Model

The **Black-Scholes Model** [BS73] is the first well-known model for option pricing. Given a *volatility*  $\sigma_B$ , the evolution of the *forward price* at time  $t$ ,  $\hat{F}_t$ , is modeled as:

$$d\hat{F}_t = \sigma_B \hat{F}_t dW$$

Where  $dW$  is a *Brownian motion* and  $\hat{F}_0$  is today's forward price.

In other words, the change in the forward price is a stochastic process that depends on the volatility of the option  $\sigma_B$ , which is *constant* for any value of the strike and the expiry.

But direct observation of financial markets teaches us that *volatility does depend on the strike and the expiry*.

The **Stochastic- $\alpha\beta\rho$  (SABR) Model** [HKLW02] is a model for pricing options created to overcome the limits of the Black-Scholes model.

Instead of assuming that volatility is a constant, the SABR approach assumes that also the volatility evolves according to a stochastic process that depends on a parameter  $\nu$ , which can be interpreted as the 'volatility of volatility'.

The defining equations of the model will therefore be:

$$d\hat{F}_t = \sigma_t \hat{F}_t^\beta dW_1$$

$$d\sigma_t = \nu \sigma_t dW_2$$

Where  $dW_1, dW_2$  are two *Brownian motions* correlated as follows:

$$dW_1 dW_2 = \rho dt$$

And  $\sigma_0 = \alpha$  and  $\hat{F}_0$  is today's forward price.

## The SABR Model iii

Given today's forward price  $f$  and a strike  $K$ , the implied volatility  $\sigma(f, K)$  in the SABR model is given by:

$$\sigma(f, K) = \frac{\alpha}{(fK)^{(1-\beta)/2} \left\{ 1 + \frac{(1-\beta)^2}{24} \log^2 f/K + \frac{(1-\beta)^4}{1920} \log^4 f/K + \epsilon_1 \right\}} \cdot \frac{z}{x(z)} \\ \cdot \left\{ 1 + \left[ \frac{(1-\beta)^2}{24} \frac{\alpha^2}{(fK)^{1-\beta}} + \frac{1}{4} \frac{\rho\beta\nu\alpha}{(fK)^{(1-\beta)/2}} + \frac{1-3\rho^2}{24} \nu^2 \right] t_{ex} + \epsilon_2 \right\} \quad (1)$$

Where  $\epsilon_1, \epsilon_2 \approx 0$  as they contain a series of terms whose value can be neglected.

With:

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

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



## The SABR Model $\nu$

Lastly, it is worth highlighting that in the special **at-the-money (ATM)** case,  $f = K$ , (1) becomes:

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

Note that the above equation yields the exact same result as the main equation of the SABR model in the  $f = K$  case, yet it provides a computationally faster option for the ATM scenario; so we will implement both in our final model.

# Calibrating a SABR Model i

**Calibrating** a SABR model means to find the set of parameters  $\alpha, \beta, \rho, \nu$  that best explain the observed market volatilities. This is usually done with constrained optimization algorithms such as **Differential Evolution** [SP97] or **Sequential Least Squares Programming**.

The constraints for the SABR parameters are:

1.  $\alpha > 0$  - because it is a volatility, thus greater than 0;
2.  $0 \leq \beta \leq 1$  - since  $\beta < 0$  means that high forwards barely change over time, and  $\beta > 1$  implies wild changes in the variation of the forward, we constrain its values;
3.  $-1 \leq \rho \leq 1$  - because it is a correlation value;
4.  $\nu > 0$  - because it is the volatility of the volatility.

# Calibrating a SABR Model ii

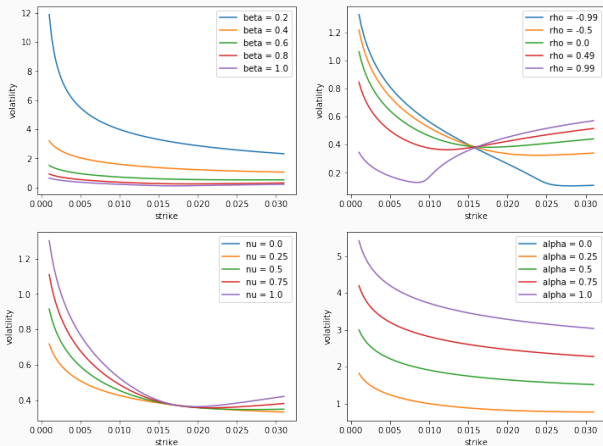


Figure 1: Analysis of parameter variation given

$$\alpha = 0.11, \beta = 0.7, \rho = -0.27, \nu = 0.81.$$

In their 2002 paper, Hagan et al. show how  $\beta$  and  $\rho$  play very similar roles in determining the shape of the volatility smile, as they both affect the *skew* of the curve, i.e. the slope of the implied volatility with respect to the strike  $K$ .

The suggestion of the authors was to either determine the value of  $\beta$  from a *log-log regression* of the past values of the observed  $\sigma_{ATM}$  against the forward  $f$  or by simple *a priori* considerations, and then undertake the problem of estimating  $\alpha, \rho, \nu$  keeping  $\beta$  fixed.

## The Dataset

---

# Structure of the data

Each cell in our dataset stores the observed market volatilities for 36 different swaptions, according to the following structure:

1. rows:

- 1.1 36 **swaptions** with different combinations of *tenor* and *expiry*, each one with its own **forward rate (fwd)**;
- 1.2 **tenor** in [2, 10, 15, 30] + [3.75];
- 1.3 **expiry** in [0.25, 0.5, 0.75, 1, 2, 5, 10] + [2.2];

2. columns:

- 2.1 the **market volatilities for strike spreads (in bps)** represent the values in *base points* of the forward minus the strike,  $f - K$ , for a given swaption. There is a total of 9 columns: [-150, -100, -50, -25, 0, 25, 50, 100, 150], where spread = 0 represents the ATM scenario in which  $f = K$ . For each one of the 36 swaptions, we will be able to extract the 9 corresponding strikes via  $f + \text{spread} * 0.0001$ .

			Market volatils. for strike spreads (in bps)					
Tenor	Expiry	Fwd	-150	-100	-50	-25	0	25
2	0.25	.0107	<b>0.00</b>	1.05	0.48	0.43	0.43	0.42
2	0.50	.0111	<b>0.00</b>	0.96	0.51	0.46	0.45	0.44
2	0.75	.0116	<b>0.00</b>	0.82	0.50	0.46	0.45	0.44
2	1.00	.0122	<b>0.00</b>	0.68	0.48	0.45	0.45	0.44
2	2.00	.0162	<b>0.00</b>	0.91	0.54	0.46	0.45	0.45
2	5.00	.0284	0.40	0.35	0.32	0.31	0.30	0.30
2	10.0	.0339	0.30	0.27	0.25	0.24	0.23	0.23

## Calibrating the Model

---



The provided  $36 \times 12$  dataframe gives us explicit information about the *observed market volatilities* (our  $y$ , in econometric terms), and implicit information about the *strike rates* of each swaption (our  $x$ ).

To extract the  $36 \times 12$  dataframe in which each cell is a strike rate, we simply apply the following formula to each cell:  $f + spread * 0.0001$ , where  $f$  and  $spread$  are the corresponding row (fwd) and column (strike spread) values for that specific cell. A snippet of the output of this process is shown in the next slide.

## Preprocessing ii

			Strike rates for strike spreads (in bps)					
Tenor	Expiry	Fwd	-150	-100	-50	-25	0	25
2	0.25	.0107	<b>-.004</b>	.001	.006	.008	.011	.013
2	0.50	.0111	<b>-.004</b>	.001	.006	.009	.011	.014
2	0.75	.0116	<b>-.003</b>	.002	.007	.009	.012	.014
2	1.00	.0122	<b>-.003</b>	.002	.007	.01	.012	.015
2	2.00	.0162	.001	.006	.011	.014	.016	.019
2	5.00	.0284	.013	.018	.023	.026	.028	.031
2	10.0	.0339	.019	.024	.029	.031	.034	.036

## Dealing with negative rates i

The first four rows of the previous slide highlight a problem that started affecting SABR models especially after the financial crisis of 2008: **negative rates**. Since the SABR model implicitly assumes rates to be strictly positive, we need to adopt a modified version of the model.

### Shifted SABR

The **Shifted SABR** model is the most common tweak to overcome this issue and simply consists in shifting  $f$  and  $K$  (and thus the volatility smile) by an arbitrary value  $s > 0$ .

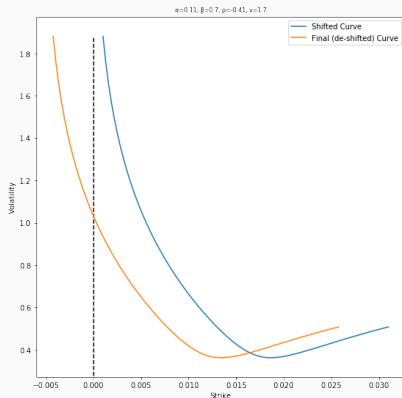
## Dealing with negative rates ii

Under the **Shifted SABR**, our forward and volatility evolution equations get a slight twist:

$$d(\hat{F}_t + s) = \sigma_t(\hat{F}_t + s)^\beta dW_1$$

$$d\sigma_t = \nu\sigma_t dW_2$$

After having estimated the coefficients, we simply shift back  $f$  and  $K$  by  $s$ , so that we have a volatility estimate for the negative rate.



# Implementation i

The implementation was done in Python3.

1. The first step consisted in implementing equations (1) and (2) in a function `SABR(alpha, beta, rho, nu, F, K, time)` that estimates the implied market volatility for a given value of the strike  $K$ .

2. Following one of the calibration methodologies quoted by Gauthier [GR09], I created from scratch a **differential evolution algorithm** using numpy only:

```
DiffEvol(object_function, bounds, *args)
```

Given an input function and a set of boundaries, differential evolution returns the set of constrained parameters that minimize the output of the input function.

## Implementation ii

3. The input function for our Differential Evolution algorithm is the RSS, where each residual is given by the estimated volatility, i.e. `SABR(alpha, beta, rho, nu, F, K, time)`, minus the observed volatility.
4. to calibrate each one of the 36 swaption with the least amount of code, I wrapped the above functions in a class:

```
SABR_swaption(F,      # forward rate : float
               K,      # strikes : iterable
               time,   # expiry (in yrs) : float
               vols,   # mkt volatilities : iterable
               calibration="DE_homebrew",
               beta=0.7)
```

## Choosing $\beta$

- As mentioned at the end of our SABR model discussion,  $\beta$  is to be chosen *before* the other model parameters are estimated. Choosing  $\beta$  along with  $\alpha, \rho, \nu$  has the potential downside of overfitting market noise, as Hagan suggested.
- Since our data is cross-sectional and thus *time-static*, also the option of going for a *log-log regression* of the past values of the observed  $\sigma_{ATM}$  against the forward  $f$  is to be discarded.

I experimented with the different *a priori* values of  $\beta$  suggested by Hagan et al., namely (0, 0.5, 1), obtaining very unstable values for the other parameters. Following West [Wes05], I ended up finding very stable parameters and good fit using a value of  $\beta = 0.7$ .

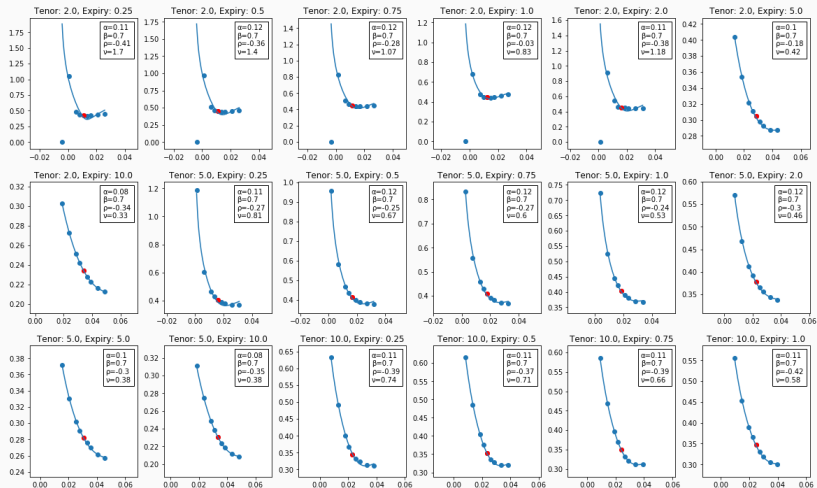
## Results Showcase

---

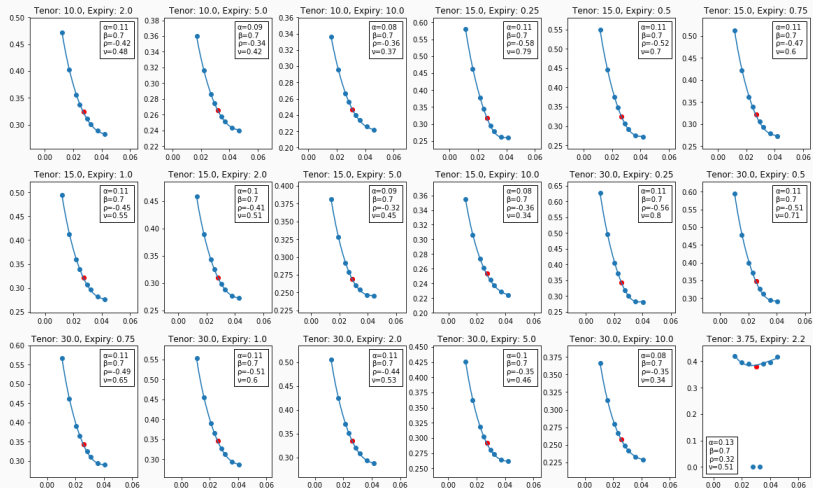


Estimation of  $\alpha, \rho, \nu$  parameters was conducted holding fixed  $\beta = 0.7$  to avoid fitting market noise. Estimation was done via Differential Evolution using a very conservative number of epochs (6000), population (40), and mutation rate (dithering in  $[1.0, 1.5]$ ).

# Results Showcase ii



# Results Showcase iii



The average error in predicted volatility per observed unit is:

$$\sum_{i=1}^{36} \left\{ \sum_{j: \sigma_{K_j} \neq 0}^J \frac{|\epsilon_{i,j}|}{36J} \right\} = 0.0066$$






Which is evidence of almost-perfect interpolation of market data by our SABR model.

Detailed information regarding model coefficients, errors, and predicted volatilities are presented in the attached .csv files.

Some comments on our results:

1. For all swaptions (except the last), we have  $\rho < 0$ . This means that the Wiener Processes that govern the evolution of the forward and the volatility are inversely correlated. In other words, higher forwards imply lower volatility.
2. Except the last one, no other swaption shows the least amount of volatility in correspondence of  $K = f$ , like it happens in the traditional volatility smile scenario. This means that there are intrinsic biases in the market that are compensated by a curve that has its minimum for  $K > f$ .

# References i

-  Fischer Black and Myron Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economy **81** (1973), no. 3, 637–654.
-  Pierre Gauthier and Pierre-Yves Henri Rivaille, *Fitting the smile, smart parameters for sabr and heston*.
-  Patrick Hagan, Deep Kumar, Andrew Lesniewski, and Diana Woodward, *Managing smile risk*, Wilmott Magazine **1** (2002), 84–108.
-  Rainer Storn and Kenneth Price, *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization **11** (1997), no. 4, 341–359.
-  Graeme West, *Calibration of the sabr model in illiquid markets*, Applied Mathematical Finance **12** (2005), no. 4, 371–385.

Thank you!