SABR MODEL IN ORE

Abstract. Description of SABR model implementation in ORE.

Contents

1. SABR Model

1.1. **Specification.** The SABR model is given by (see e.g. [3])

$$
\begin{aligned}
dF &= \sigma(F+d)^{\beta}dW_1 && (1.1)\\
d\sigma &= \nu\sigma dW_2 && (1.2)\\
dW_1dW_2 &= \rho dt && (1.3)\\
F(0) &= F_0 && (1.4)\\
\sigma(0) &= \alpha && (1.5)
\end{aligned}
$$

with

- F a forward rate or price, e.g. a forward swap rate, a forward libor or overnight term rate, with initial value F_0
- d a (non-negative) displacement; in the classic set up we have $d = 0$, for $d > 0$ we allow for negative F down to $-d$
- σ the stochastic forward rate volatility
- W_1, W_2 Wiener processes
- α the initial value of σ
- $\beta \in [0, 1]$ the exponent for the forward
- $\nu > 0$ the volatilitiy of σ
- ρ the correlation between forward and volatility, given as a constant, instantaneous correlation between the driving Wiener processes

The latter 4 parameters are referred to as the "SABR parameters" of the model.

Date: 30 January 2026.

1.2. **Resolution.** There are different ways to obtain European option prices or implied Black / Bachelier volatilities. In the following we use

- Hagan2002Lognormal: This is formula 2.17a from [3], implemented in QuantLib as unsafeSabrLognormalVolatility().
- Hagan2002Normal: This is formula B.64 from [3], implemented in QuantLib as unsafeSabrNormalVolatility().
- Hagan2002NormalZeroBeta: This is for $\beta = 0$, formula B.59a with $C(f) = 1$, implemented in QuantExt as normalSabrVolatility().
- Antonov2015FreeBoundaryNormal: This is for $\beta = 0$. It is from [2] and [1] for the premium calculation, using [4] for the conversion to implied volatility. Implemented in QuantExt as normalFreeBoundarySabrPrice() and normalFreeBoundarySabrVolatility(), the latter using bachelierBlackFormulaImpliedVol() from QuantLib.
- KienitzLawsonSwaynePde: This is an adaption of [5], with kind permission of the author published in QuantExt as KienitzLawsonSwayneSabrPdeDensity().
- FlochKennedy: This exposes sabrFlochKennedyVolatility() from QuantLib. Note: This variant shows unstable results and is considered experimental. More work is needed.

1.3. **Domains and preferred output types.** The resolution methods support strike domains and have preferred output types as listed in table 1.3. The output type can be converted from the preferred type to any other type in a numerical post-processing step, but each method has a natural, primary output type.

Method	Strike Domain	Preferred Output Type
Hagan2002Lognormal	$[-d, \infty)$	Shifted Lognormal Volatility
Hagan2002Normal	$[-d, \infty)$	Normal Volatility
Hagan2002NormalZeroBeta	$(-\infty, \infty)$	Normal Volatility
Antonov2015FreeBoundaryNormal	$(-\infty, \infty)$	Premium
KienitzLawsonSwaynePde	$[-d, \infty)$	Premium
FlochKennedy	$[-d, \infty)$	Shifted Lognormal Volatility

TABLE 1. Strike domains and preferred output types.

1.4. **Imply α.** The parameter α can be implied from the at-the-money volatility during model calibration. This is done

- using a closed-form solution for Hagan2002NormalZeroBeta implement in QuantExt normalSabrAlphaFromAtmVol().
- using a zero-search with the Brent solver for all other resolution methods

1.5. **Calibration.** The calibration of the SABR parameters is performed on the market smile points, which are given by

- option expiry and swap underlying length in the case of interest rate swaptions
- option expiry in the case of interest rate cap/floors

For each market smile point the free SABR parameters are calibrated to quotes of the given market smile. If the market smile quotes are not given in the preferred output type of the resolution method, we convert the market quote to the preferred output type first.

The atm volatility used to imply alpha (if applicable) is linearly interpolated from the given market smile. In general, no interpolation is necessary since a market quote for the atm strike is usually given.

If no free parameters are given, we use the given initial SABR parameters unchanged for the given market point, except for α is this is implied.

If at least one free parameter is given we run an optimization using the LevenbergMarquardt optimizer on the following target function f.

$$T(x) = \sum_{i=1}^{n} \frac{m_i - s(x, K_i)}{m_r} \tag{1.6}$$

with

- $x = (x_0, x_1, x_2, x_3)$ the input SABR parameters
- m_i the *ith* market quote, converted to the preferred output type of the resolution method
- $s(x, K)$ the output of the resolution method for parameters x and strike K
- m_r the reference market quote used to compute the relative error. This is generally set to the maximum over all market quotes, becase far otm premium quotes can be close to zero.

In fact, the optimization is run on transformed parameters to be able to run an unconstrained optimization. The transformation from transformed parameters x to actual SABR parameters is defined as follows:

$$\beta = \max(\epsilon_1, \min(1-\epsilon_1, e^{-x_1^2})) \tag{1.7}$$

$$\alpha = \max(\epsilon_1, e^{-x_0^2} m_v / \max(f+d, \epsilon_1)^\beta) \tag{1.8}$$

$$\nu = \max(\epsilon_1, e^{-x_2^2} m_\nu) \tag{1.9}$$

$$\rho = 1_{|x_3|<2.5\pi} \epsilon_2 \sin(x_3) + \epsilon_2 (1_{x_3>2.5\pi} - 1_{x_3<-2.5\pi}) \tag{1.10}$$

The inverse transformation is given as:

$$x_1 = \sqrt{-\ln(\min(1-\epsilon_1, \max(\epsilon_1, \beta)))} \tag{1.11}$$

$$x_0 = \sqrt{-\ln(\min(1-\epsilon_1, \max(\epsilon_1, \alpha \max(f+d, \epsilon_1)^\beta / m_v)))} \tag{1.12}$$

$$x_2 = \sqrt{-\ln(\min(1-\epsilon_1, \max(\epsilon_1, \nu / m_\nu)))} \tag{1.13}$$

$$x_3 = \arcsin(\max(-\epsilon_2, \min(\epsilon_2, \rho))) \tag{1.14}$$

Here, $\epsilon_1 = 10^{-7}$, $\epsilon_2 = 1 - 10^{-4}$, $m_v = 0.02$, $m_\nu = 2.0$. Furthermore, f denotes the forward atm rate, d the displacement here.

The optimization is run from user provided initial SABR parameters. If the target function T on this first run is below an user provided "early exit" error threshold t_A, i.e.

$$T(x) < t_A$$

the result of the optimization will be accepted. Otherwise more optimization runs will be performed using random initial guesses generated from a Halton sequence. Let (h_0, h_1, h_2, h_3) be a point of the halton sequence. Then the start value for the optimization will be set to (in actual parameter space)

$$\alpha = (\epsilon_1 + 0.01h_0)/\max(f + d, \epsilon_1)^\beta \tag{1.15}$$
$$\beta = \epsilon_1 + h_1\epsilon_2 \tag{1.16}$$
$$\nu = \epsilon_1 + m_\nu h_2 \tag{1.17}$$
$$\rho = \epsilon_2(2h_3 - 1) \tag{1.18}$$

If for any of the subsequent optimization runs, the error lies below t_A the corresponding minimum will be accepted. If for a given maximum number of runs N the error never lies below the given threshold, the best found solution with target value $T(x_b)$ is compared against the maximum acceptable threshold t_B and if

$$T(x_b) < t_B$$

the solution x_b is accepted. Otherwise the calibration is marked as "unsuccessful". After all market points have been calibrated we have either a 1D array (cap volatilities) or a 2D matrix (swaption volatilities) of calibrated SABR parameters with possibly missing entries from unsuccessful calibrations. As a last step, these missing values are interpolated linearly (1D case) or using Laplace Interpolation in 2D where the x and y axis are indexed by the option time to maturity and underlying swap length (i.e. the interpolation grid is not equidistant in general). The interpolated values are updated if not admissiable for the SABR model, i.e.

$$\alpha \to \max(0, \alpha) \tag{1.19}$$
$$\beta \to \max(0, \min(1, \beta)) \tag{1.20}$$
$$\nu \to \max(0, \nu) \tag{1.21}$$
$$\rho \to \max(-1, \min(1, \rho)) \tag{1.22}$$

1.6. **Interpolation.** SABR parameters for option expiry times / underlying swap lengths that do not lie on the market quotation grid will be interpolated linearly (1D case for Cap/Floors) or bilinearly (2D case for swaptions) with flat extrapolation.

2. Parametrization in ORE

For details on the parametrization see also the ORE User Guide. This documentation focusses on the link between the parametrization and methodology explained in 1.

The SABR interpolation is specified as

```
<StrikeInterpolation>Hagan2002NormalZeroBeta</StrikeInterpolation>
```

for cap/floor volatility surfaces and

```
<Interpolation>Hagan2002NormalZeroBeta</Interpolation>
```

for swaption volatility surfaces. In both cases the *Input* volatility type is given by

```
<VolatilityType>Normal</VolatilityType>
```

The displacement d applied in the SABR *model* formulation is either read from the input market quotes, or can be overridden with

```
<ModelShift>0.0</ModelShift>
```

for cap/floors (single displacement d for the whole surface) resp.

```
<ModelShift>0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0</ModelShift>
```

for swaptions (each underlying swap length has its own associated displacement d). In the following

```
<ParametricSmileConfiguration>
```

the initial values for the optimization for each parameter are specified, either as a single value that is used for all options or individually for the array of option expiries (caps), matrix of swaption expiries / underlying swap lengths, e.g.

```
<Parameters>
  <Parameter>
    <Name>alpha</Name>
    <InitialValue>0.0050</InitialValue>
    <Calibration>Implied</Calibration>
  </Parameter>
  <Parameter>
    <Name>beta</Name>
    <InitialValue>0.0</InitialValue>
    <Calibration>Fixed</Calibration>
  </Parameter>
  <Parameter>
    <Name>nu</Name>
    <InitialValue>0.30</InitialValue>
    <Calibration>Calibrated</Calibration>
  </Parameter>
  <Parameter>
    <Name>rho</Name>
    <InitialValue>0.0</InitialValue>
    <Calibration>Calibrated</Calibration>
  </Parameter>
</Parameters>
```

The “Calibration” field is used to distinguish

- “Fixed”: fixed values that are not calibrated, but prescribed by the user resp. set to a value corresponding to the model flavour (e.g. $\beta = 0$ for Hagan2002NormalZeroBeta and Antonov2015FreeBoundaryNormal).
- “Implied”: if α is implied from atm market volatility
- “Calibrated”: takes part in optimization

It follows a Calibration section with the maximum number of calibration attempts N, the early exit error threshold t_A and the maximum acceptable error threshold t_B:

```
<Calibration>
  <MaxCalibrationAttempts>10</MaxCalibrationAttempts>
  <ExitEarlyErrorThreshold>0.005</ExitEarlyErrorThreshold>
  <MaxAcceptableError>0.05</MaxAcceptableError>
</Calibration>
```

Finally, the *output* volatility type and displacement d can be overridden. By default, input volatility type and displacement is used. In the case of swaptions an override would look like this

```
    <OutputVolatilityType>Normal</OutputVolatilityType>
    <OutputShift>0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0</OutputShift>
```

References

[1] Antonov, A., Konikov, Spector: SABR spreads its wings, Risk, July 2013
[2] Mixing SABR models for negative rates, September, 2015, https://ssrn.com/abstract=2653682
[3] Hagan, P.: Managing Smile Risk, 2002
[4] Jaeckel, P.: Implied Normal Volatility, 2017, http://www.jaeckel.org/ImpliedNormalVolatility.pdf
[5] Kienitz, J.: VBA Code to create a SABR density with PDE methods, 2017