New Stochastic Volatility Models

- PDE, Approximation, Deep Pricing and Calibration -
Webinar - WBS Quantshub, 14.05.2020

Jörg Kienitz - Finciraptor, UCT, AMNA, Quaternion

UCT, BUW, Finciraptor finciraptor.de, joerg.kienitz@finciraptor.de

Disclaimer

This presentation and any accompanying material are being provided solely for information and general illustrative purposes. The author will not be responsible for the consequences of reliance upon any information contained in or derived from the presentation or for any omission of information therefrom and hereby excludes all liability for loss or damage (including, without limitation, direct, indirect, foreseeable, or consequential loss or damage and including loss or profit and even if advised of the possibility of such damages or if such damages were foreseeable) that may be incurred or suffered by any person in connection with the presentation, including (without limitation) for the consequences of reliance upon any results derived therefrom or any error or omission whether negligent or not. No representation or warranty is made or given by the author that the presentation or any content thereof will be error free, updated, complete or that inaccuracies, errors or defects will be corrected.

The views are solely that of the authors and not of any affiliate institution. The Chatham House rules apply.

The presentation may not be reproduced in whole or part or delivered to any other person without prior permission of the author.

The presentation is based on:
M. Felpel, J. Kienitz, T. McWalter,
Effective Stochastic Volatility: Applications to ZABR-type Models, [8]

J. Kienitz, S. K. Acar, Q. Liang, N. Nowaczyk,
The CV Makes the Difference - Control Variates for Neural Networks, [19]

SV and ML

Outline

Outline

Discrete Volatility Surfaces

We start with the *discrete implied volatility surface*.

- $\mathcal{T} := \{T_1, T_2, \ldots, T_N\}$ as set of option maturities
- $\mathcal{K} := \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_N\}$, $\mathcal{K}_i := \{K_{i,1}, K_{i,2}, \ldots, K_{i,M_i}\}$ be sets of strike values indexed by the number of maturities considered. Usually $\mathcal{K}_i = \mathcal{K}_j$ for all $1 \leq i, j \leq N$.
- The implied volatility for each quoted option with respect to $T_i \in \mathcal{T}$, $K_j \in \mathcal{K}$:

$$\begin{aligned} \Sigma_d : \mathcal{T} \times \mathcal{K} &\to \mathbb{R}^+ \\ (T, K) &\mapsto \sigma_d. \end{aligned}$$

- The map Σ_d is called the discrete implied volatility surface.
- The implied volatilities are with respect to a reference model (e.g. Bachelier, Black-Scholes-Merton)

Continuous Volatility Surface

For practical purposes we consider the *continuous implied volatility surface* given by the map

$$\Sigma_{c,0} : [0, T] \times [K_l, K_u] \to \mathbb{R}^+$$
$$(T, K) \mapsto \sigma_c$$

- Many approaches for modeling the dynamics of (inst.) volatility and to determine Σ_c wrt a reference model exist
- We consider stochastic volatility models (SVM).
- Selecting SVM and its parameters determine Σ_c (and its dynamics $\Sigma_{c,t}(T, K)$, $t \in \mathbb{R}^+$).
- Matching to the observed discrete implied volatility surface is called calibration, and, once a model is calibrated, the continuous implied volatility surface may be used for interpolation and extrapolation.

General Stochastic Volatility Models

In particular, we consider the model GSVM determined by the coupled SDEs given by

$$\begin{cases} d\tilde{F}_t = C(\tilde{F}_t) v_t \, dW_t^{(1)}, & \tilde{F}_{t_0} = f, \\ dv_t = \mu(v_t)\, dt + \nu(v_t)\, dW_t^{(2)}, & v_{t_0} = \alpha, \\ \text{with} \quad d\langle W^{(1)}, W^{(2)} \rangle_t = \rho \, dt. \end{cases} \tag{1.1}$$

Our general framework provides to this approach covers most well known stochastic volatility models: SABR model (including displacements), [13], free SABR (fSABR) model, [3], ZABR model, [2], Stein-Stein model, [22], Schoebel-Zhu model, [21], Heston model, [16].
But also new variants of the classic models including fZABR (free ZABR), mrZABR (mean reverting ZABR) or fmrSABR (free mean reverting SABR).

Classic and New Stochastic Volatility Models

Specific choices of the coefficients lead to ZABR-type models. In particular we consider GSVMs of the form (1.1) where the functions μ, C and ν are of the form given by:

μ	C	ν	Model
0	$\tilde{F}_t^{\beta}$	v_t	SABR
0	1	v_t	Normal SABR (nSABR)
0	$(\tilde{F}_t + d)^{\beta}$	v_t	Displaced SABR
0	$\lvert\tilde{F}_t\rvert^{\beta}$	v_t	Free SABR (fSABR)
$\kappa(\theta - v_t)$	$\tilde{F}_t^{\beta}$	v_t	mean reverting SABR (mrSABR)
0	$\tilde{F}_t^{\beta_1}$	$v_t^{\beta_2}$	ZABR
0	$(\tilde{F}_t + d)^{\beta_1}$	$v_t^{\beta_2}$	Displaced ZABR
0	$\lvert\tilde{F}_t\rvert^{\beta_1}$	$v_t^{\beta_2}$	Free ZABR (fZABR)
$\kappa(\theta - v_t)$	$\tilde{F}_t^{\beta_1}$	$v_t^{\beta_2}$	mean reverting ZABR (mrZABR)

General Stochastic Volatility Models

- The choice of model and parameters should ensure the best fit to the current (discrete) market implied volatility surface
- The dynamics are suitable for risk management and trading of exotic contracts.
- Sometimes ease of implementation determines the choice of the model, rather than model suitability.
- We provide a general modeling approach with a tractable computational framework that does not require this compromise.

Outline

Effective PDE

To achieve numerical tractability, we use *singular perturbation methods* to derive an approximate PDE, called the *effective PDE*, for the marginal probability density of the asset. Here, this probability density should be understood as

$$\mathbb{P}\left[F < \tilde{F}_t < F + dF \,\middle|\, \tilde{F}_{t_0} = f, v_{t_0} = \alpha\right].$$

This technique was originally introduced by Hagan *et al.* [10, 9, 11] for SABR models.

Effective PDE

Given the reduced density for a specified exercise time T,

$$Q(t,F)\,dF = \mathbb{P}\Big[F < \tilde{F}_t < F + dF \,\Big|\, \tilde{F}_{t_0} = f, v_{t_0} = \alpha\Big]. \qquad (1.2)$$

we can then recover option prices with payoff h by an evaluation of

$$V_{h,Q}(T,K) = \int h(F)Q(T,F)\,dF$$

To compute the reduced density, we derive a PDE of the form

$$\partial_t Q(t,F) = \partial_{FF}\left[D(t,F)Q(t,F)\right], \quad Q(t_0,f) = \delta(F-f), \quad (1.3)$$

where $D(\cdot,\cdot)$ is a function that involves the model parameters and depends on t and the asset value F. It can be viewed as a local volatility function.

Effective PDE

- The effective PDE, also called the effective forward equation, is accurate to order $\mathcal{O}(\varepsilon^2)$.
- For achieving a stable and efficient numerical implementation to solve the PDE, we especially need to specify the boundary behavior.
- This leads us to consider two PDEs for accumulating probability, with the probability densities (for lower and upper bound) are denoted by Q^L and Q^R.

$$\partial_t Q^L(t) = \lim_{F \downarrow b_l} \partial_F \left[D(t,F) Q(t,F) \right], \qquad Q^L(t_0) = 0$$

and $$\partial_t Q^R(t) = -\lim_{F \uparrow b_u} \partial_F \left[D(t,F) Q(t,F) \right], \quad Q^R(t_0) = 0.$$

Effective PDE

Theorem 1

Let GSVM (1.1) *obey Assumptions I–IV (appendix), an effective PDE for the effective probability* (1.2) *and* (1.3)*, is given by:*

$$D(t,F) = \frac{1}{2}a(t)^2 C(F)^2 e^{G(t)}\left(1 + 2b(t)z(F) + c(t)z(F)^2\right),$$

where the coefficients are specified as

$$a(t) = Y(t, t_0, \alpha), b(t) = \frac{1}{a(t)s(t)} I_1(t),$$

$$c(t) = b(t)^2 + \frac{1}{a(t)s(t)^2} I_2(t) - \frac{6b(t)}{s(t)^2} I_3(t) + \frac{2}{a(t)s(t)^2} I_4(t),$$

$$G(t) = -s(t)c(t) - s(t)b(t)\Gamma_0 + \frac{1}{a^2} I_5(t), \Gamma_0 = -C'(f).$$

Effective PDE

- The general expressions for I_1 to I_4 can be found in the Appendix,
- Time-dependent parameters can be handled
- We use also can apply effective parameters (Parameter Averaging, see [1] or [12]

Implied Volatility Formulae using Effective Parameters

For a fixed maturity T we derive effective parameters for GSVM:

$$\bar{b} = \frac{2}{T^2}\int_0^T ub(u)\,du$$

$$\bar{c} = \frac{3}{T^3}\int_0^T u^2 c(u)\,du + \frac{18}{T^3}\int_0^T b(u)\int_0^u vb(v)\,dv\,du - 3\bar{b}^2$$

$$\bar{G} = \frac{1}{T}\int_0^T G(u)\,du + \frac{1}{T}\int_0^T u(c(u)-\bar{c})\,du.$$

These constant parameters are used to determine effective SABR parameters:

$$\nu_{\text{eff}} = \sqrt{\bar{c}}, \qquad \rho_{\text{eff}} = \frac{\bar{b}}{\sqrt{\bar{c}}}, \qquad \alpha_{\text{eff}} = \alpha\left(1 + \frac{1}{2}\bar{G} + \frac{1}{4}\alpha\bar{b}\Gamma_0 T\right). \quad (1.4)$$

$\Rightarrow$ Allows to approximate implied volatility to order $\mathcal{O}(\varepsilon^2)$ with SABR approximation formula!

Example: ZABR

For the ZABR model the explict parameters are:

$$\begin{aligned}
\nu_{\text{eff}} &= \nu \alpha^{\beta_2 - 1} \sqrt{1 + (\beta_2 - 1)\rho^2} \\
\rho_{\text{eff}} &= \frac{\rho}{\sqrt{1 + (\beta_2 - 1)\rho^2}} \\
\alpha_{\text{eff}} &= \alpha \left(1 + \frac{1}{4} \rho^2 \nu^2 \alpha^{2(\beta_2 - 1)} (1 - \beta_2)\, T \right)
\end{aligned} \tag{1.5}$$

To guarantee that the term $\sqrt{D(t, F)}$ remains real, we further impose the condition

$$\beta_2 > 1 + \frac{\rho^2 - 1}{\rho^2}.$$

Example: mrZABR

Considering the mean-reverting ZABR model with a reversion back to the initial state, i.e., $\theta = \alpha$:

$$b(t) = \frac{\rho\nu\alpha^{\beta_2-2}}{\kappa(t-t_0)}\left(1-e^{-\kappa(t-t_0)}\right),$$

$$\begin{aligned} c(t) = {} & \frac{(1+\rho^2)\nu^2\alpha^{2(\beta_2-2)}}{\kappa^2(t-t_0)^2}\left(1-e^{-\kappa(t-t_0)}\right)^2 \\ & + \frac{6\rho^2\nu^2\alpha^{2(\beta_2-2)}}{\kappa^3(t-t_0)^3}\left(1-e^{-\kappa(t-t_0)}\right)\left(1-\kappa(t-t_0)-e^{-\kappa(t-t_0)}\right) \\ & + (1+\beta_2)\frac{2\rho^2\nu^2\alpha^{2(\beta_2-2)}}{\kappa^2(t-t_0)^2}\left(1-(1+\kappa(t-t_0))e^{-\kappa(t-t_0)}\right), \end{aligned}$$

$$G(t) = -\alpha^2(t-t_0)c - \frac{\rho\nu\alpha^{\beta_2}}{\kappa}\left(1-e^{-\kappa(t-t_0)}\right)\Gamma_0 + \frac{\nu^2\alpha^{2(\beta_2-1)}}{2\kappa}\left(1-e^{-2\kappa(t-t_0)}\right).$$

Example: mrZABR

Fixing a specified maturity T, the corresponding constant effective parameters are:

$$\bar{b} = \frac{2\rho\nu\alpha^{\beta_2-1}}{\kappa^2 T^2}\left(\kappa T - 1 + e^{-\kappa T}\right),$$

$$\begin{aligned}\bar{c} = {} & \frac{3(1+\rho^2)\nu^2\alpha^{2(\beta_2-1)}}{2(\kappa T)^3}\left(2\kappa T + 4e^{-\kappa T} - 3 - e^{-2\kappa T}\right) \\ & +6(1+\beta_2)\frac{\rho^2\nu^2\alpha^{2(\beta_2-1)}}{(\kappa T)^3}\left(\kappa T + 2e^{-\kappa T} - 2 + \kappa T e^{-\kappa T}\right) \\ & -12\rho^2\nu^2\alpha^{2(\beta_2-1)}\left(\frac{\kappa T - 1 + e^{-\kappa T}}{(\kappa T)^2}\right)\end{aligned}$$

$$\bar{G} = \frac{\nu^2\alpha^{2(\beta_2-1)}}{4\kappa^2 T}\left(2\kappa T + e^{-2\kappa T} - 1\right) - \frac{1}{2}\bar{c}\,T - \frac{\rho\nu\alpha^{\beta_2}}{\kappa^2 T}\left(\kappa T - 1 + e^{-\kappa T}\right)\Gamma_0.$$

Outline

Examples for Probability Distributions

Figure 1 shows the output obtained by numerically solving the effective PDE. It is the density of the asset at maturity and depends on all the input parameters.

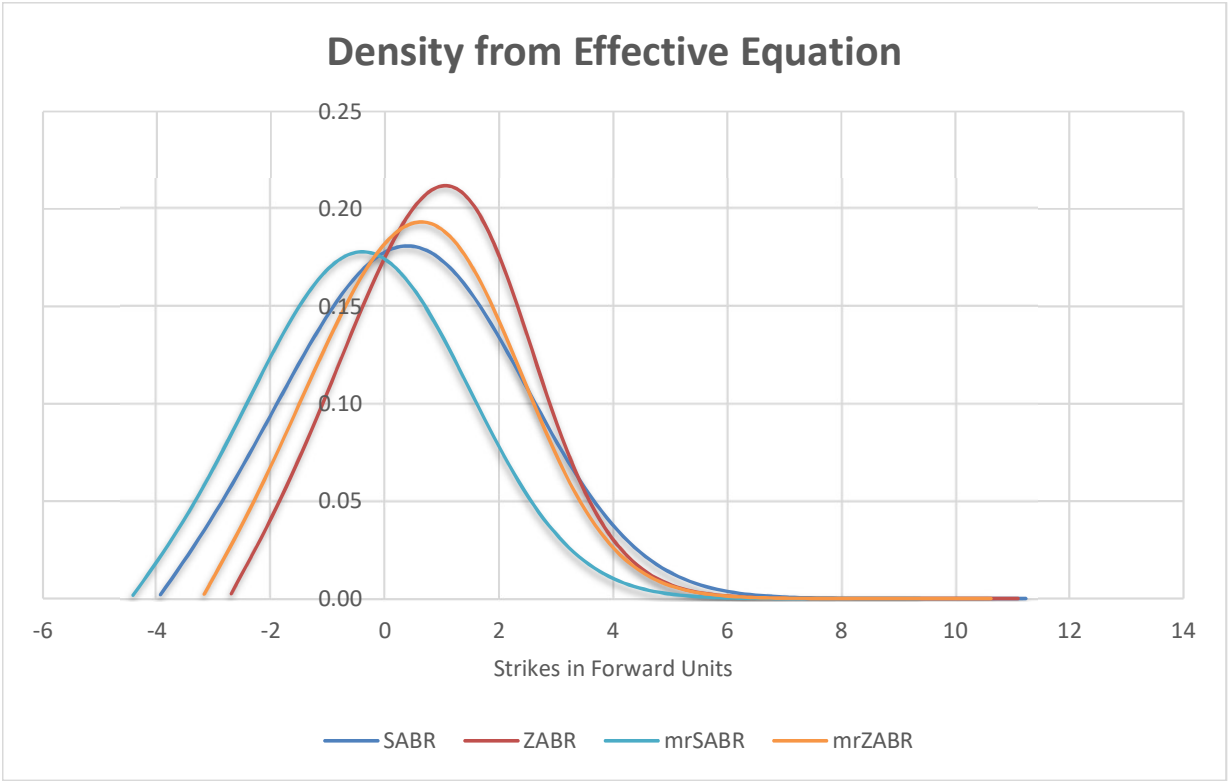


Figure: Output from numerically solving the effective PDE for SABR, ZABR, mrSABR and mrZABR.

Examples of Implied Bachelier Volatilities

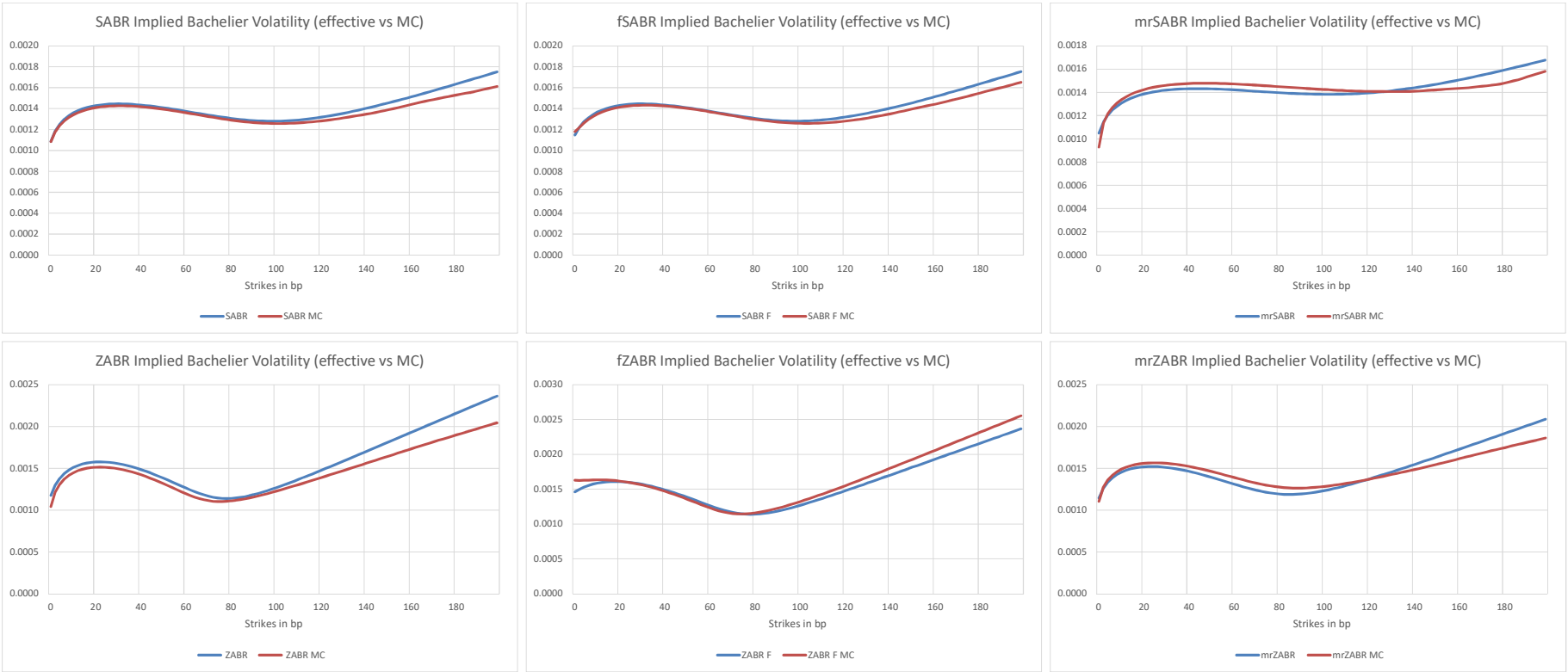
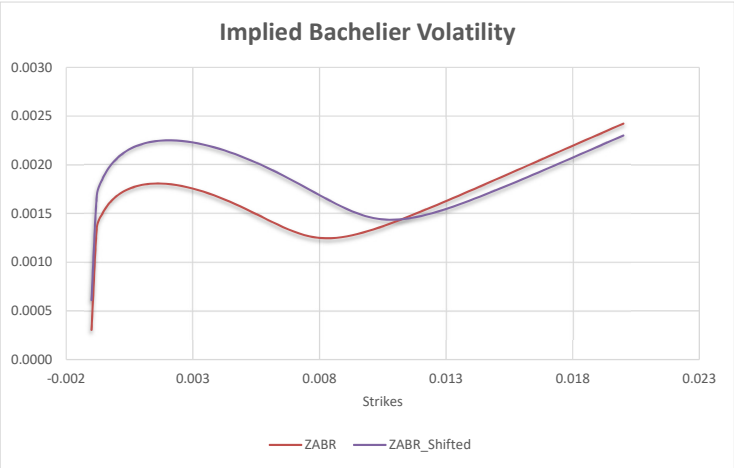


Figure: Implied Bachelier volatility computed from the Call option prices obtained from the effective equation and Monte Carlo simulation for the SABR (top left), ZABR (top right), mrSABR (mid left), mrZABR (mid right), fSABR (bottom left) and fZABR (bottom right).



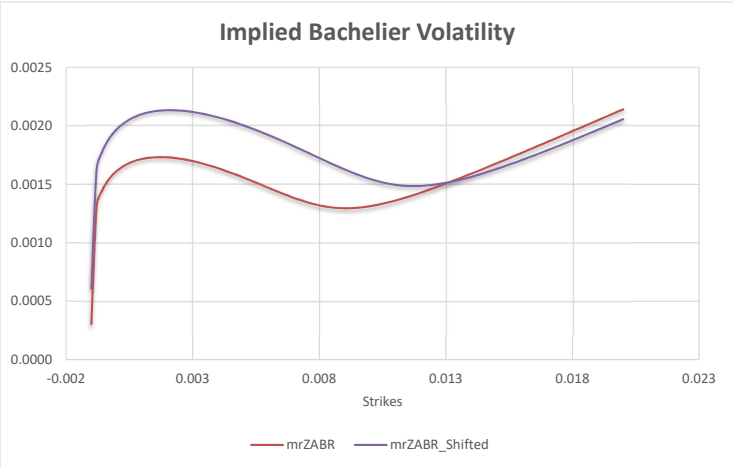


Figure: The implied volatility for the ZABR model with parameters $\beta = 0.5$, $\beta_2 = 0.8$, $\nu = 0.3$, $\rho = -0.8$, an underlying forward rate of 0.005, which is shifted by 0.002, and a displacement of 0.001 (left) and mrZABR with mean reversion of $\kappa = 0.2$ and shift (right).

ZABR with different CEV parameters (volatility)

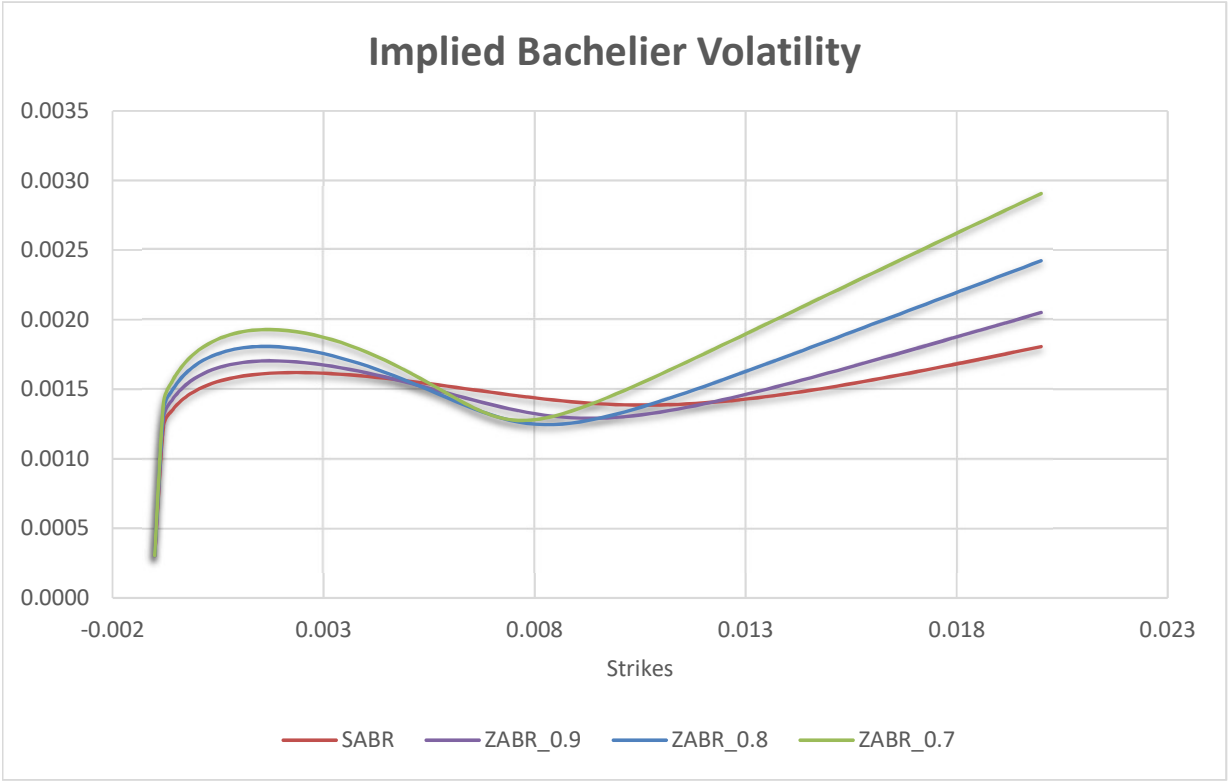


Figure: Implied volatility for the ZABR model when β_2 changes.

mrZABR with different reversion rates

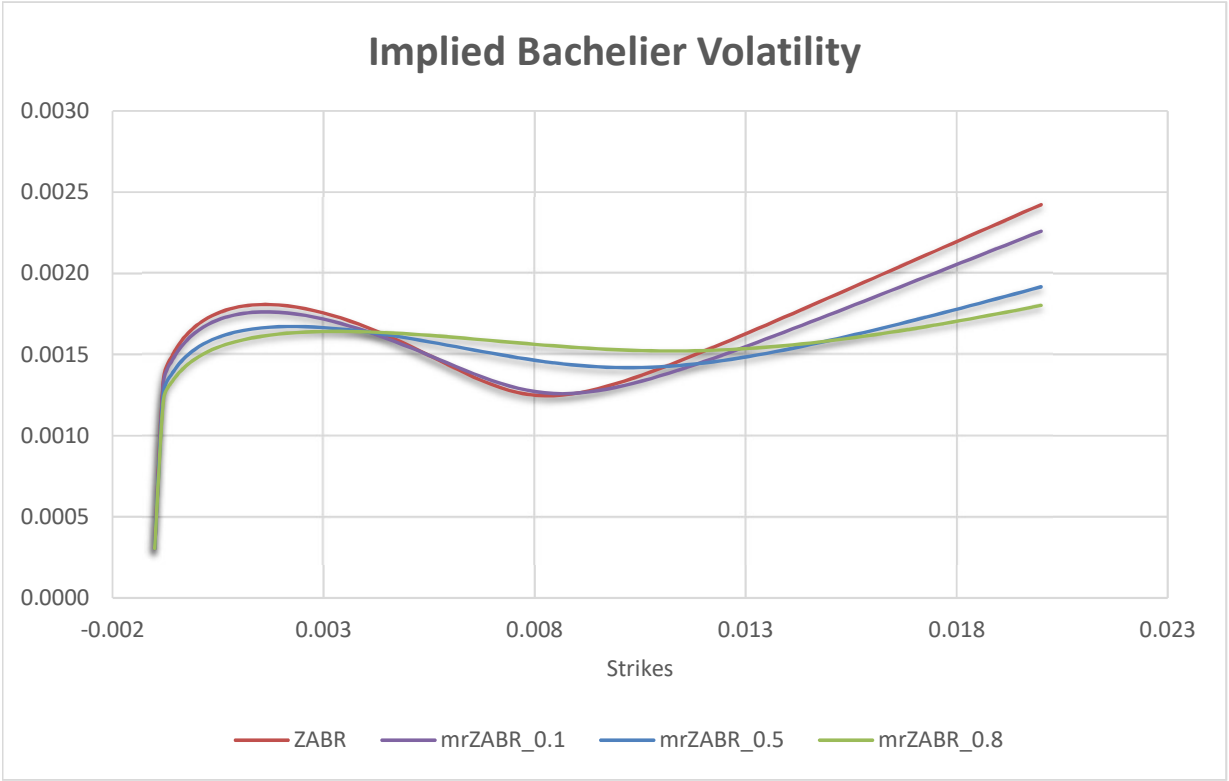


Figure: Implied volatility for the mean reversion ZABR when κ changes.

ZABR Density with different CEV parameters (volatility)

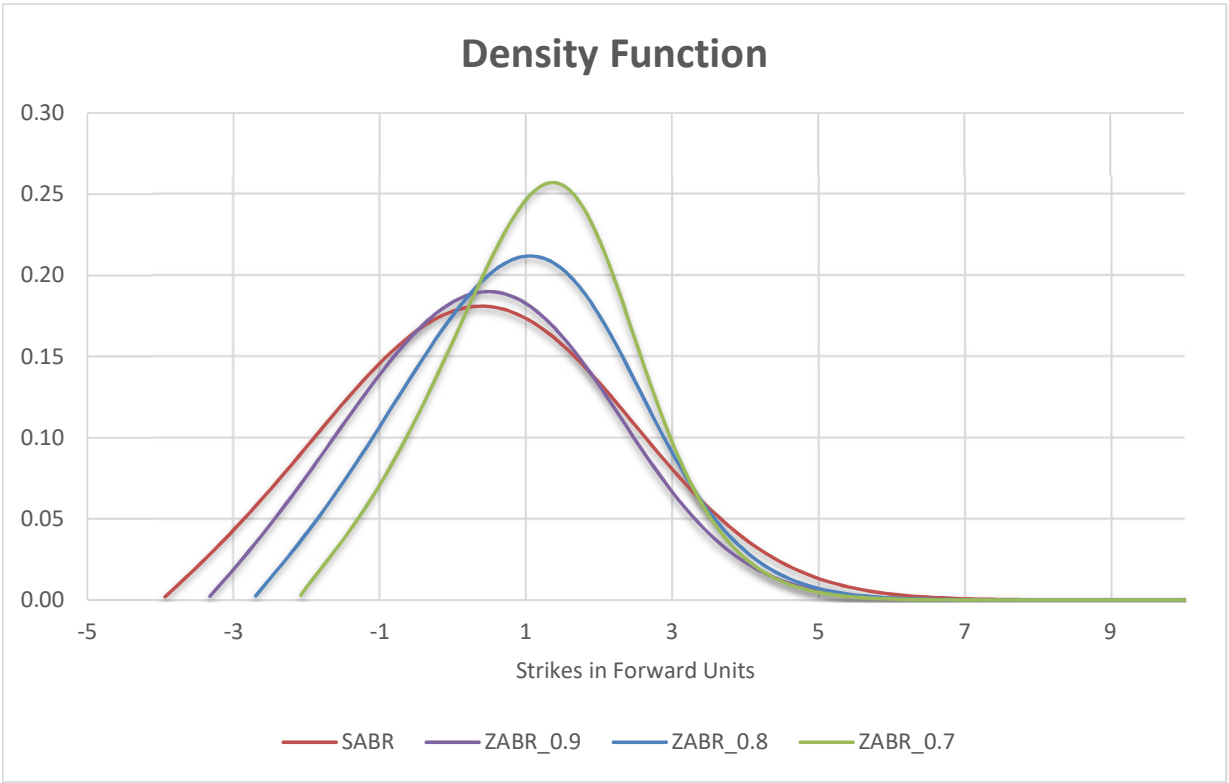


Figure: Density for the ZABR model when β_2 changes. This model may lead to higher and steeper peaks in the density function, compared with the SABR model.

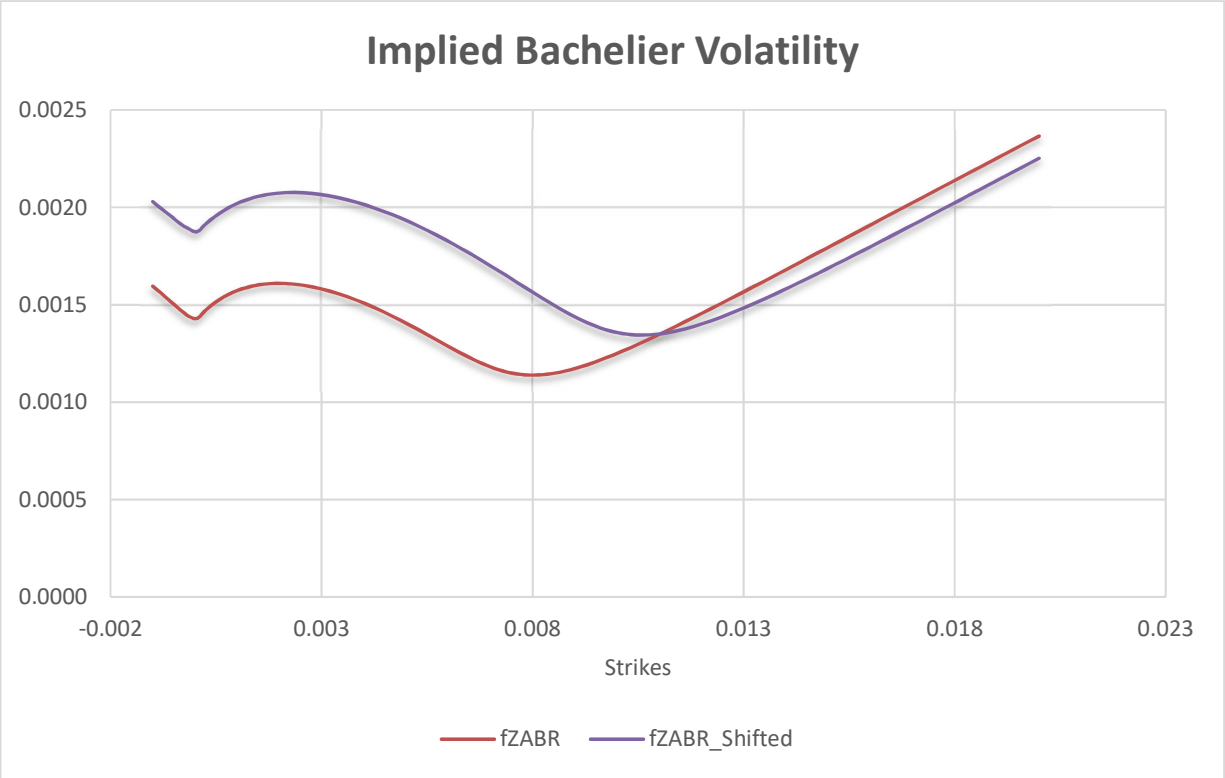


Figure: Free ZABR implied volatility with parameters as in Figure 3.

Outline

Outline

Supervised Learning

The setting we consider is the *Supervised Learning* approach where we consider a set

$$\mathcal{D} := \{(x_1, y_1), \ldots, (x_n, y_n)\},$$

with $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^l$ for $i = 1, \ldots, n$.

- $\mathcal{X} := (x_1, \ldots, x_n)$ the *inputs* and $\mathcal{Y} := (y_1, \ldots, y_n)$ the *targets* or *labels*.
- The dimension l is the dimension of the targets.
- We realize the supervised learning using an deep neural network approach.

Outline

The neural network

- The neural network we use consists of one input layer ($\#nodes = d$), with d being the input tensor dimension.
- The output layer's number of nodes corresponds to the dimension l of the labels y quantities that we wish to learn.
- We stack a number of hidden layers l_i, $i = 1, \ldots, n$ on top of the input layer each having n_i number of nodes.
- For each layer we specify an activation function (*elu* for our experiments) and linear activation for the output.

The Training

- We create the data ($\approx$ 40.000 samples) for SABR, eg. using

```
alphaMin = 0.01;  alphaMax = 0.3 ; alphaDelta = alphaMax - alphaMin;
betaMin = 0.01; betaMax = 0.4; betaDelta = betaMax - betaMin;
nuMin = 0.01; nuMax = 0.25 ; nuDelta = nuMax - nuMin;
rhoMin = -.9; rhoMax = -0.5 ; rhoDelta = rhoMax - rhoMin;
TMin = 1; TMax = 2.5; TDelta = TMax - TMin;
```

- The strike values are set to

```
strikes = np.linspace(kmin,kmax,Nk)
```

- The data set is being scaled
- For the training we split the sets $\mathcal{X}, \mathcal{Y}$ into training $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$ and validation (aka test) sets $(\mathcal{X}_{\text{val}}, \mathcal{Y}_{\text{val}})$.
- 15% of the data for creating the validation set.
- we choose a cost function that measures the difference of the ANN's output to the true values $\mathcal{Y}_{\text{train}}$
- The optimization is done using the Adam optimizer, [20].

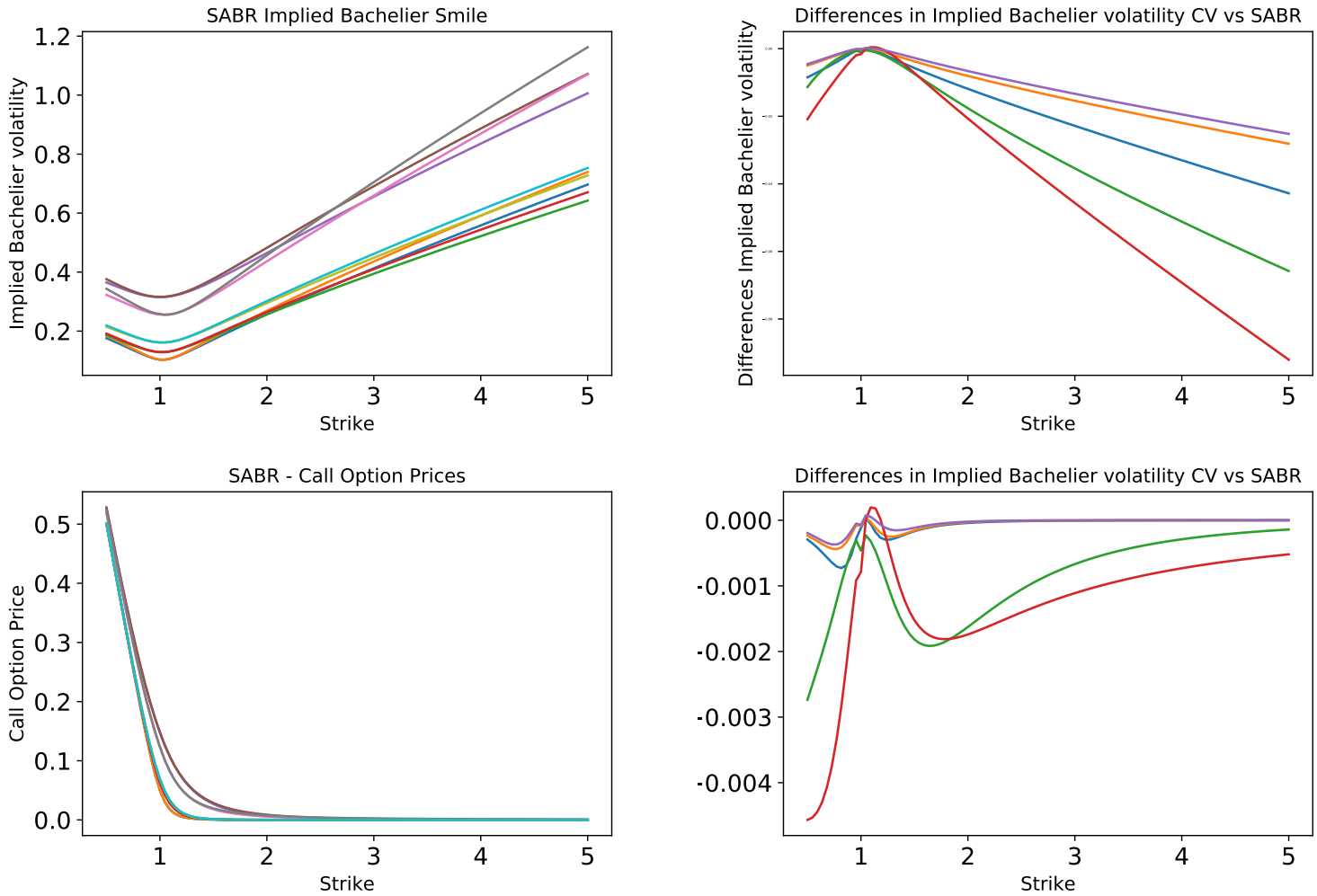


Figure: (Left) Some realizations for implied Bachelier volatilities / prices computed by PDE and approximation formula (Right) Differences of the methods.

The testing phase

- For testing we consider the model's learning history.
- If the results are not satisfactory we may alter the topology of the network by adding or subtracting layers, resp. nodes.
- General methods including cross-validation or over-fitting issues are not repeated here and can be found in [14].
- Finally, the result is applied on newly generated data that is neither used during training nor validation.

Outline

The Control Variate Approach

- Applying a control variate approach to Deep Learning we consider another set $\mathcal{Y}_{\text{CV}}$ and set $\mathcal{Y}_{\text{new}} := \mathcal{Y} - \mathcal{Y}_{\text{CV}}$.
- We apply the training to $(\mathcal{X}, \mathcal{Y}_{\text{new}})$. It remains to choose the set $\mathcal{Y}_{\text{CV}}$.
- Once having learned the relationship between $\mathcal{X}$ and $\mathcal{Y}_{\text{new}}$ we use the ANN to predict values. To this end let x_{input} be the input, y_{predict} the prediction derived by applying the ANN and y_{cv} the value derived by applying the control variate.
- Then, we derive an approximation to true value by setting

$$y_{\text{true}} \approx y_{\text{cv}} + y_{\text{predict}}$$

- We use the loss function for 'extreme' cases to match the 'true' solution.

The Control Variate Approach

The CV method can be used in two ways. To this end let n be the network:

- Learn in a region R where the chosen CV is reasonable good using the objective function (empirical risk):

$$l_1(y, n(x)) = \|y - n(x)\|_2^2,\ x \in R$$

- Learn in a region R where the chosen CV is reasonable good and outside the region $x \notin R$ using the objective function (regularized empirical risk):

$$l_2(y, n(x)) = 1_{\{x \in R\}} l_1(y, n(x)) + \beta 1_{\{x \notin R\}} \|A \cdot (\tilde{y}_{\text{cv}}(x) - (y_{\text{cv}} + n(x)))\|_2^2,$$

  A a linear map, β a scalar and $\tilde{y}_{\text{cv}}$ a control variate for $x \notin R$.
- The second term is a regularization and we may take $A = Id$.

The Control Variate Approach

The CV method can be used in two ways:

- 'Cheap to compute' CV + learn the difference (Difference Learning). Final price: CV + learned differences within a training region. Then, use penalty on the loss function for 'extreme' cases to control the behaviour of the approximation outside a given region. In this way we aim to use the exact numerical method (PDE, Integration, etc.) as CV outside the training region to stabilize the approximation and use a simple deep learning architecture - feedfwd, fully connected.
- Choose a (possible expensive to compute) CV to determine the asymptotics and learn the difference (Asymptotic Learning). Suggested in [4]. The aim is to control the asymptotic behaviour. They suggest methods for achieving that (Spline methods and Constrained Radial Layers). This needs sophisticated deep learning architecture.

Control Variates

In practice we often face situations where for a given model reasonably good approximation formulae exist.

(i) Special case of a model, e.g. SVM with $\rho = 0$ or r an all purpose approximation taking into account all model parameters

(ii) different model, e.g. Black-Scholes model for computing Heston model prices, [16].

(iii) Standard contracts close to an exotic, e.g. Bermudan swaptions, see for instance [18].

(iv) Markov projection for baskets of SVM or LV.

These are exactly the cases where the control variate can be applied in the sense of difference learning.
Instead of learning the values of the set $\mathcal{Y}$ we only learn the labels $\mathcal{Y}_{\text{new}}$ calculated by applying the approximation, analytic or vanilla pricing, ie. the control variate $\mathcal{Y}_{\text{CV}}$.

Calibration with ANN

Using ANN to calibrate models we have different choices

- Inverse map approach
  - Learn the inverse map from observable market data.

    This approach needs a lot of observed market data for training. This might be a bottleneck.
  - Learn the inverse map from model prices (training + validation) and use observable market data as test input.

    Possibility to have as many training data we want. Learning the inverse map directly may suffer from instabilities, see [15, 7].
- Two-step approach
  Learn the pricing and calibrate using standard techniques using the learnd pricing.

  Possible to create as many samples for training/validation as we like using some pricing function.

  Separation of pricing and calibration leads to stability. Calibration is lightning fast, see [6, 5, 17]

Calibration with ANN

Applying a two step ML approach we see the advantages:

- **Independence of the pricing approximation**
  For each model the most favourable pricing approximation could be used. Since the generation of prices is separated from the actual calibration we even can rely on Monte Carlo methods.
- **Availability of training data**
  It is possible to generate as many training data as we like. We could also use different price approximations, eg. net architectures for different parameter sets.
- **Interpretatbility**
  The interpretability of the results is the same as in the classic approach. Since we work with models instead of purely ANN based methods the model parameters have the same meaning as in the classic approach. The ANN is nothing but a complex Black-Box approximation that we need to assure it works.

We directly apply the improved pricing methodology using the CV since for calibration the optimizer calls the trained ANN pricing function.

Outline

The Neural Network

The deep neural network for the examples we choose:

```
import keras
from keras.layers import Activation
from keras import backend as K
from keras.utils.generic_utils import get_custom_objects


keras.backend.set_floatx('float64')
input1 = keras.layers.Input(shape=(Nparams,))              # input layer

x1 = keras.layers.Dense(20,activation = 'elu')(input1)     # hidden layer 1
x2 = keras.layers.Dense(20,activation = 'elu')(x1)         # hidden layer 2
x3 = keras.layers.Dense(20,activation = 'elu')(x2)         # hidden layer 3

x4=keras.layers.Dense(Nk,activation = 'linear')(x3)        # output layer; size depends on option surface

# set up the model
modelGENp = keras.models.Model(inputs=input1, outputs=x4)
modelGENp.summary()
```

We use *only* 40.000 samples.

The SABR Model - Training

- We train two neural networks on varying parameters for α, β, ν, ρ and T keeping the forward equal to 1 (wlog due to transformation properties of the SABR model).
- We train on a log-moneyness range from -0.5 to 1.5.
- One is the standard approach without using a control variate and the other is with applying the SABR approximation formula for Bachelier volatility as control variate.
- The resulting errors and the standard deviation are much smaller for the latter case!

We have $e_{\text{abs, std}} = 0.00373$ and an average error of $e_{\text{av,std}} = 4.17e-05$ and $e_{\text{abs,cv}} = 0.00034$ as well as $e_{\text{av,std}} = 3.99e-06$. The absolute error is 10 times smaller for the control variates technique.

SABR Model - Learning Call Prices

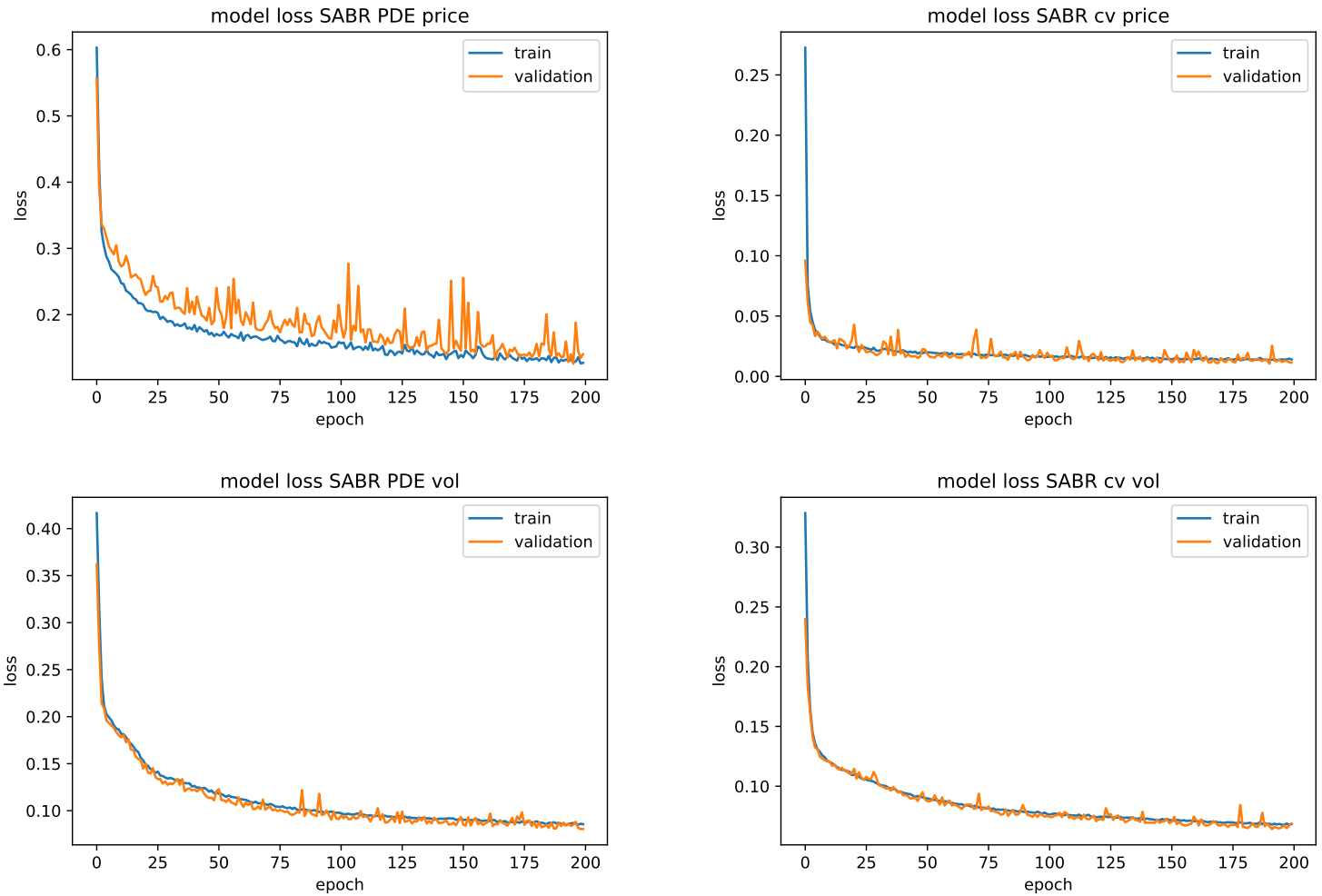


Figure: Learning history for PDE prices/implied Bachelier volatilities. (Left) Standard (Right) CV.

The SABR Model - Learning Call Prices

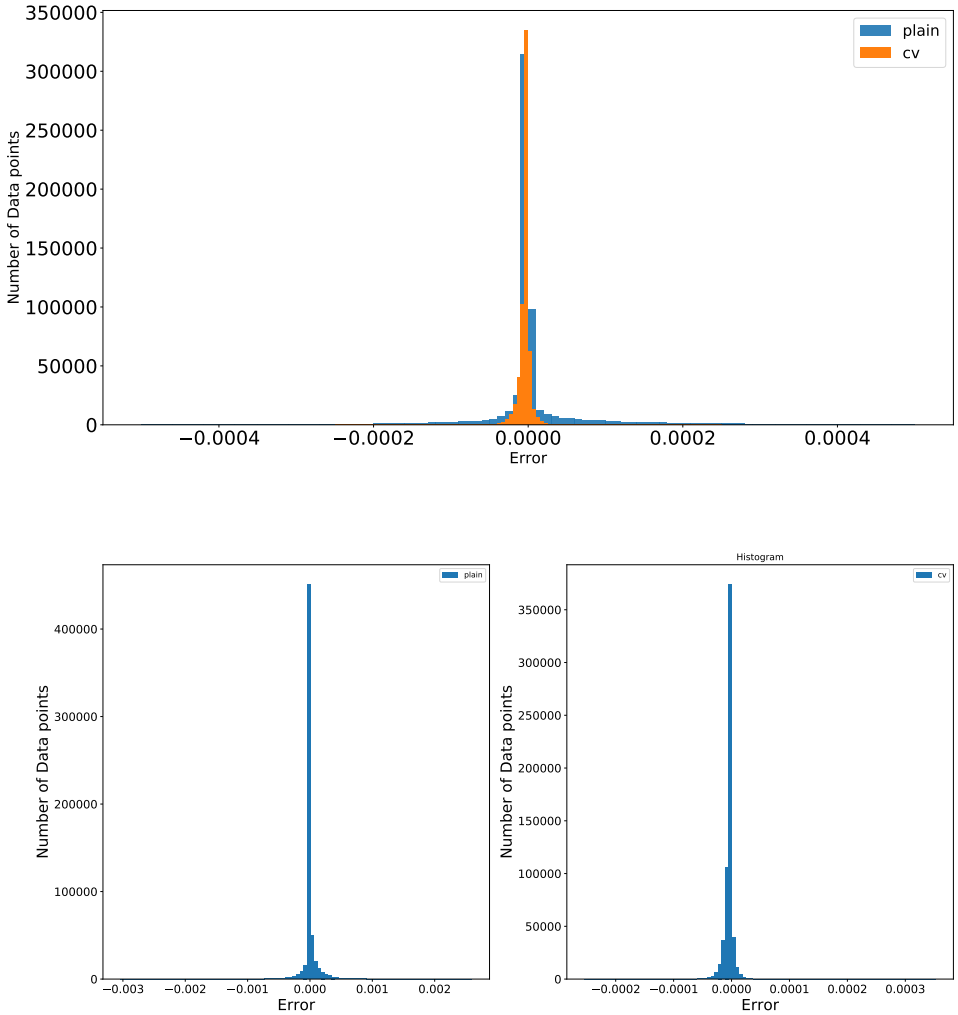


Figure: (Top) Error histogram for CV and Standard appoach (Bottom) Histograms for CV and Standard approach.

The SABR Model - Learning Implied Volatilities

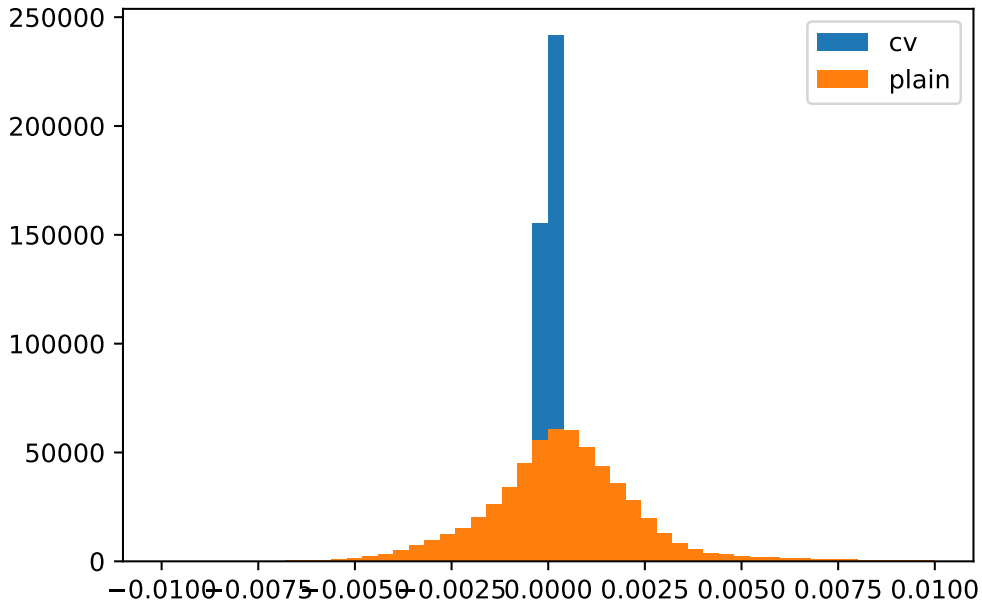


Figure: For a given maturity we show the relative error for the standard (blue) and the control variate (orange) approach. The errors are calculated along the strike range of moneyness from 0.5 to 1.5 for implied volatilities.

The SABR Model

To further illustrate the superiority of the control variate approach we consider the relative errors for a given maturity for moneyness levels from 0.5 to 1.5.

Figure: For a given maturity we show the relative error for the standard (orange) and the CV (blue) approach along the strike range of moneyness from 0.5 to 1.5.

SABR Model - Learning Call Prices

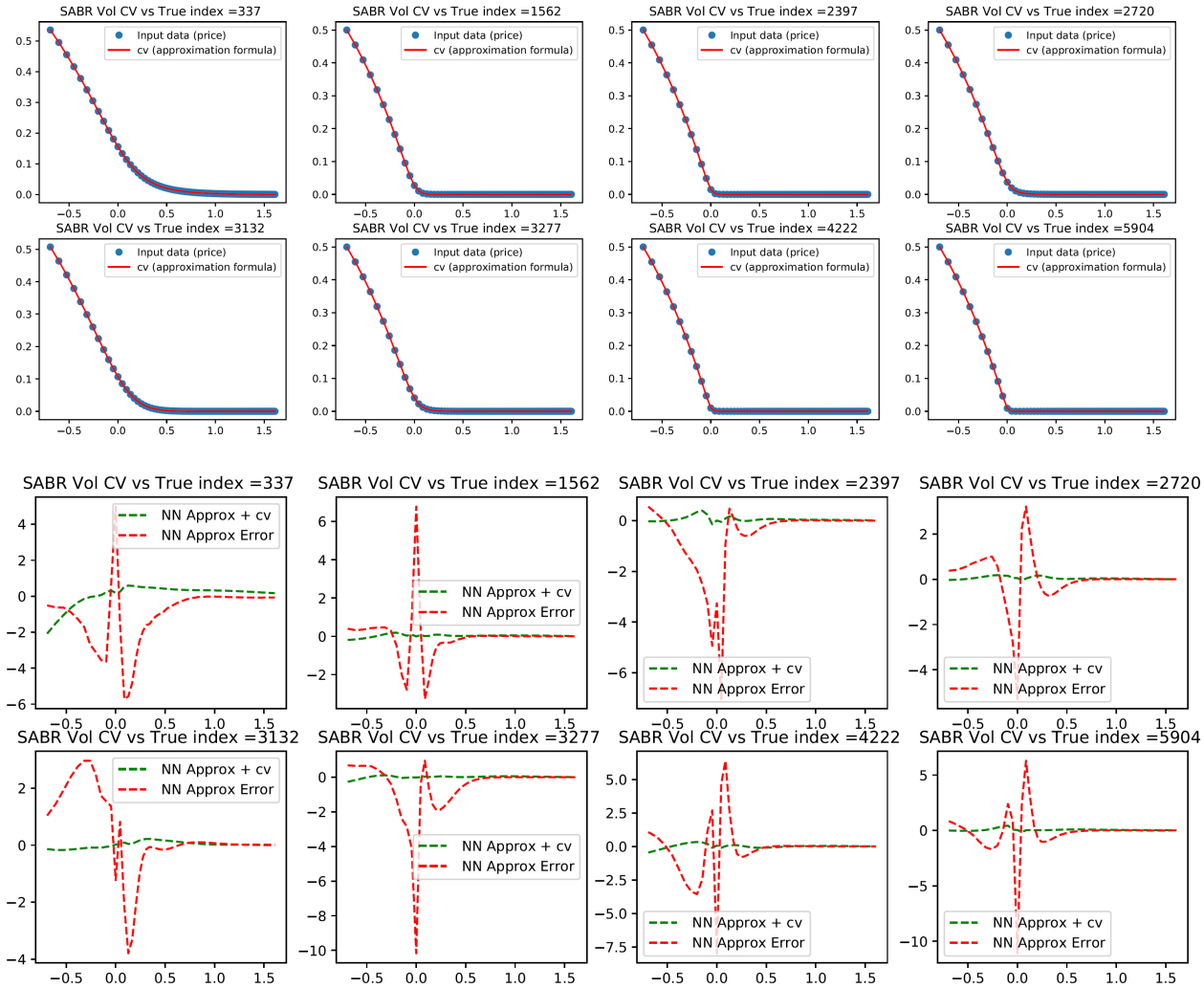


Figure: Trained neural net with CV applied to unseen data. (Top) Prices, (bottom) Differences in bp.

Pitfalls

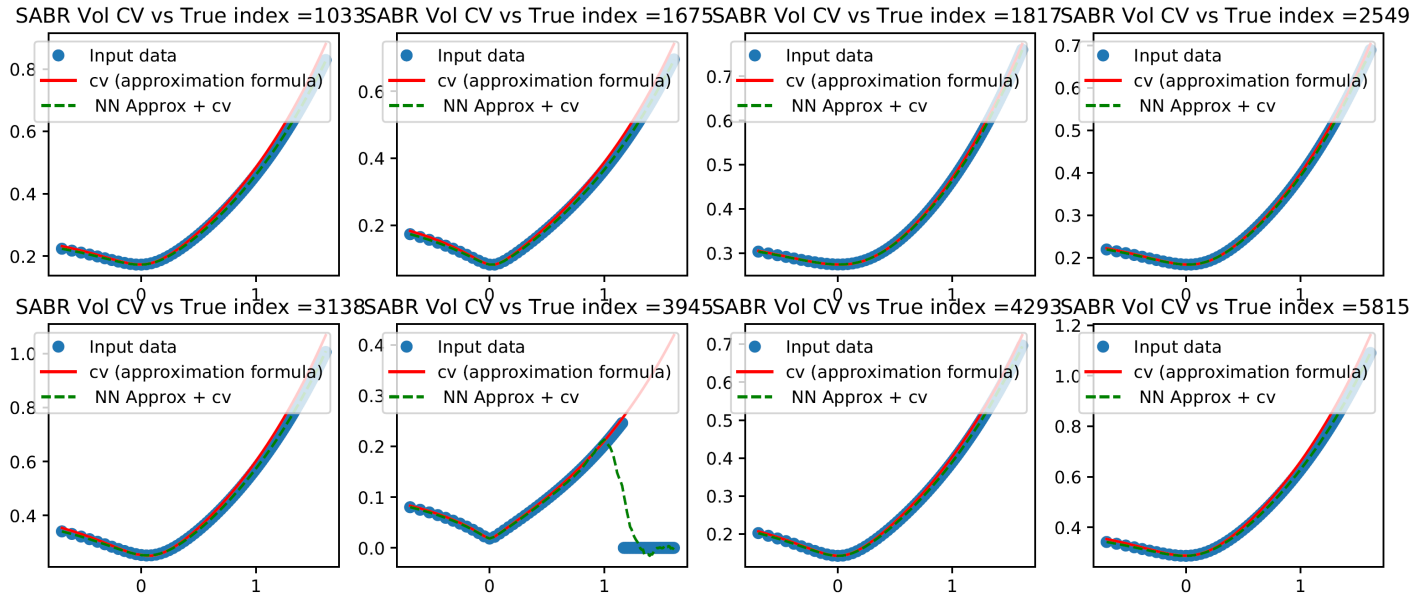


Figure: Trained neural net with CV applied to unseen data.

- The CV approach does not automatically account for data failures when learning the pricing function.
- If there is erroneous data the ANN is trained as if the data is correct.
- To this end we used some corrupted data when training the pricing within a SABR model.

Summary and Conclusions

- General Stochastic Volatility models (GSVM) is considered
- New variants of classical models were considered
- PDE and approximation methods were introduced and illustrated with examples
- Neural networks (ANN) with Control Variates (CV) are considered
- The CV method stabilizes the ANN approach for pricing and calibration
- An extensive example from the GSVM is given, the SABR model

Outline

Assumption I

To derive the effective PDE we make the following assumptions related to (1.1):

Assumption I

The drift term, $\mu(\cdot)$, is differentiable, with derivative $\mu'(\cdot)$, and a solution $Y(t, t_0, \alpha)$ to the following PDE exists:

$$\begin{cases} \partial_t Y(t, t_0, \alpha) = \mu(Y(t, t_0, \alpha)) \\ Y(t, t, \alpha) = \alpha \\ Y(t_0, t_0, \alpha) = \alpha. \end{cases}$$

Assumption II

Assumption II

The function Y is differentiable and has an inverse function $y(t_0, t, a)$ such that

$$Y(t, t_0, \alpha) = a \qquad \Leftrightarrow \qquad \alpha = y(t_0, t, a).$$

Remark 3.1

Functions $\mu(\cdot)$ allowing a closed-form solution include:

(i) for $\mu(x) = \mu$ the solution is $Y(t, t_0, \alpha) = \alpha + \mu(t - t_0)$.

(ii) for $\mu(x) = \kappa(\theta - x)$ the solution is $Y(t, t_0, \alpha) = \alpha e^{-\kappa(t-t_0)} + \theta(1 - e^{-\kappa(t-t_0)})$.

Assumption III

The functions

$$X(t, t_0, \alpha) = \partial_\alpha Y(t, t_0, \alpha),$$
$$Z(t, u) = Z(t, u, t_0, \alpha) = y(u, t, Y(t, t_0, \alpha)),$$
$$z(F) = \int_f^F \frac{1}{C(u)}\, du,$$
$$s(t) = S(t_0, t, \alpha) = \int_{t_0}^t Z(t, u, t_0, \alpha)^2\, du$$

and

$$\psi(t, u, Z) = \nu\big(Z(t, u)\big) Z(t, u) X\big(t, u, Z(t, u)\big)$$

are well defined,

Assumption III

$X\big(t,u,Z(t,u)\big)^{-1}$ exists, and the following integral functions are defined:

$$I_1(t) = \rho \int_{t_0}^{t} \psi(t,u,Z)\,du,$$

$$I_2(t) = 2\int_{t_0}^{t} \nu\big(Z(t,u)\big)^2 X\big(t,u,Z(t,u)\big)^2 \int_{u}^{t} Z(t,v)X\big(t,v,Z(t,v)\big)^{-1}\,dv\,du,$$

$$I_3(t) = \rho \int_{t_0}^{t} \psi(t,u,Z) \int_{u}^{t} Z(t,v)X\big(t,v,Z(t,v)\big)^{-1}\,dv\,du,$$

$$I_4(t) = \rho^2 \int_{t_0}^{t} \psi(t,u,Z) \int_{u}^{t} \partial_Z\Big(\psi(t,v,Z)\Big)X\big(t,v,Z(t,v)\big)^{-1}\,dv\,du,$$

$$I_5(t) = \int_{t_0}^{t} \nu\big(Z(t,u)\big)^2 X\big(t,u,Z(t,u)\big)^2\,du.$$

Assumption IV

Assumption IV

The function $C(\cdot)$ is differentiable at f, with derivative denoted by $C'(\cdot)$.

Literature I

Andersen, L. and Piterbarg, V.

Interest Rate Modeling - Volume I: Foundations and Vanilla Models.
Atlantic Financial Press, 2010.

Andreasen, J. and Huge, B. N.

Expanded Forward Volatility.
RISK, 1, 2013.

Antonov A., Konikov, M., and Spector, M.

FreeBoundarySABR.
RISK, 2015.

Antonov A., Konikov M., Piterbarg V.

Neural Networks with Asymptotics Control.
SSRN, 3585966, 2020.

Bayer C., Horvath B., Muguruza A., Stemper B., Tomas M.

On deep calibration of (rough) stochastic volatility models.
ArXiv, 2019.

Bayer C., Stemper B.

Deep calibration of rough stochastic volatility models.
ArXiv 1810.03399, 2018.

Dimitroff G., Roeder D.R., Fries C. P.

Volatility model calibration with convolutional neural networks.
Preprint, SSRN:3252432, 2018.

Felpel M., Kienitz J., and McWalter T.

Effective Stochastic Volatility - Applications to ZABR-type models.
SSRN Preprint, 2020.

Literature II


Hagan, P., Kumar, D. , Lesniewski, A. S. and Woodward, D. E.
Universal Smiles.
Wilmott Magazine, 2016.


Hagan, P., Kumar, D.,Lesniewski, A. and Woodward, D.
Arbitrage-Free SABR.
Wilmott Magazine, 1:60–75, 2014.


Hagan, P., Lesniewski, A. and Woodward, D.
Probability Distribution in the SABR Model of Stochastic Volatility.
Working Paper, http://lesniewski..us/papers/ProbDistForSABR.pdf, 2005.


Hagan, P., Lesniewski, A. S. and Woodward, D. E.
Effective Media Analysis for Stochastic Volatility Models.
Wilmott Magazine, 1:46–55, 2018.


Hagan, P.S., Kumar, D., Lesniewski A.S. and Woodward, D.E.
Managing Smile Risk.
Wilmott Magazine, 1:84–108, 2002.


Hastie T., Tibshirani R., and Friedman J.
The Elements of Statistical Learning, 2nd edition.
Springer Series in Statistics, 2008.


Hernandez, A.
Model Calibration with Neural Networks.
SSRN: https://ssrn.com/abstract=2812140, 2016.

Literature III

Heston, S.

A closed form solution for options with stochastic volatility with applications to bond and currency options.
Rev. Fin. Studies, pages 327–343, 1993.

Horvath B., Muguruza A., Tomas M.

Deep Learning Volatility.
Preprint, 2019.

Kienitz J., Acar S., Liang Q., Nowaczyk N.

Deep Option Pricing.
Machine Learning in Finance, 1, 2020(a).

Kienitz J., Acar S., Liang Q., Nowaczyk N.

The CV makes the difference.
SSRN, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3527314, 1, 2020(b).

Kingma, A. and Ba, J.

Adam: A method for stochastic optimization.
International Conference for Learning Representation (ICLR), 2015.

Schoebel, R. and Zhu, J.

Stochastic Volatility with an Ornstein Uhlenbeck Process: An Extension.
European Finance Review, 3:23–46, 1999.

Stein, E. M. and Stein, J. C.

Stock Price Distribution with Stochastic Volatility: An Analytic Approach.
Review of Financial Studies, 4:727–752, 1991.