Fast Direct Calibration of Interest Rate Derivatives Pricing Models

Luca Sabbioni
luca.sabbioni@polimi.it
Politecnico di Milano
ISI Foundation

Marcello Restelli
marcello.restelli@polimi.it
Politecnico di Milano

Andrea Prampolini
andrea.prampolini@intesasanpaolo.com
Banca Intesa Sanpaolo

ABSTRACT

To price complex derivative instruments and to manage the associated financial risk, investment banks typically model the underlying asset price dynamics using parametric stochastic models. Model parameters are calibrated by fitting cross sections of option prices on the relevant risk factors. It is fundamental for a calibration method to be accurate and fast and, to this end, Deep Learning techniques have attracted increasing attention in recent years. In this paper, the aim is to propose a Neural Network based calibration of a pricing model, where learning is directly performed on market data by using a non-trivial loss function, which includes the financial model adopted. In particular, the model chosen is the two-additive factor Gaussian Interest Rates model in a multi-curve framework calibrated on at-the-money European swaptions. The main advantage lies in the independence from an external calibrator and in the calibration time, reduced from several seconds to milliseconds, achieved by offloading the computational-intensive tasks to an offline training process, while the online evaluation can be performed in a considerably shorter time. Finally, the efficiency of the proposed approach is tested in both a single-currency and a multi-currency framework.

CCS CONCEPTS

• **Computing methodologies → Supervised learning by regression**.

KEYWORDS

Direct Financial Calibration, Artificial Neural Networks, Inverse Problem, Pricing Models

ACM Reference Format:
Luca Sabbioni, Marcello Restelli, and Andrea Prampolini. 2020. Fast Direct Calibration of Interest Rate Derivatives Pricing Models. In *ACM International Conference on AI in Finance (ICAIF '20),October 15–16, 2020, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3383455.3422534

1 INTRODUCTION

Over-the-counter interest rate derivatives (IRDs) represent one of the deepest global markets, with a notional outstanding amount estimated at around $500 trillion in 2019 [11]. Most of the liquidity is concentrated in standardized swap and swap option (or "swaption") contracts, which are simple to evaluate. However, investment banks are also active in tailor-made, more complex ("exotic") IRDs, where sophisticated mathematical models of interest rate dynamics are used to price and manage the associated financial risk.

These models are typically parametric, i.e. they depend on a set of parameters that must be optimized so that the models can achieve an accurate fit to market data (prices of swaps and liquid swaptions). This task is called calibration (or, in other frameworks, inverse problem), and is relatively time-consuming for a business where speed is of the essence. In this context, calibration approaches based on Machine Learning [3, 10, 15–17, 21, 24, 29] are attracting increasing attention as a faster alternative to more traditional root searching algorithms. Moreover, fast calibration of interest rate models is desirable for financial institutions beyond applications to exotic IRDs: interest rates determine the discounting process for computing the present value of future cash flows, and are thus a key ingredient for modelling financial claims more generally. Within our approach, the goal is to learn, through Feed Forward Neural Networks trained on historical data, a mapping from the space of swaption prices to the parameter space. In this way, as soon as new prices are available, a set of parameters close to the optimal ones is immediately provided; these new samples can also be considered to refine the mapping in an *online* setting. In a similar fashion as in [15], the main advantage is calibration speed, achieved by offloading the computational-intensive tasks to an offline training process while the online evaluation can be performed in a considerably shorter time. However, unlike [15] and other Supervised Learning techniques proposed in this context [19, 23], our approach does not consider solutions provided by other external calibrators, but relies only on market data and on the financial model taken into account, thus reducing noise for approximations. As a consequence, the target is not directly available and the loss function (Sections 2, 3) is not trivial: for this reason it differs from standard Supervised Learning techniques. As in [16], the non-convexity of the objective function leads to the presence of local minima, thus leading to the need to consider global and local optimization algorithms (Section 5).

The proposed approach is evaluated (in Section 7) in a real framework, where the chosen model is the two-additive factor Gaussian model G2++ [7], calibrated on European-style at-the-money (ATM) swaptions, both from a single-currency and from a multi-currency dataset (presented in Section 4).

2 CALIBRATION

2.1 Feedback function

In this section, we deal with the definition of the calibration problem, without considering the details of the financial model M chosen. In particular, we are interested in finding the best mapping between the market data Q and its related set of model parameters Θ. We consider the quote predicted by a pricing model as $\hat{Q}_\Theta(\tau, \xi)$, where τ is used to denote the peculiar properties that identify the chosen financial instrument (e.g., the maturity, the tenor, or the strike if an option is not ATM), and ξ refers to exogenous factors, e.g., the discount and forward curves.Often the dependence on ξ will be omitted. The daily market data, for a specific day d, consist of a set T_d of financial instruments to be calibrated. We can define the following *daily cost function*:

$$\epsilon(\Theta_d, Q_d) = \sqrt{\sum_{\tau \in T_d} \alpha_\tau (\hat{Q}_{\Theta_d}(\tau) - Q_d(\tau))^2}\,, \tag{1}$$

that is, the weighted Root Mean Squared Error between the predicted prices $\hat{Q}_d$ and the related market values Q_d, where α_τ is a specific weight chosen for every instrument τ in the set of daily instruments T_d.

The main idea is to immediately provide the set of parameters Θ_d once a new set of market values is available, without the need to restart the calibration procedure from scratch. For this reason, we considered the possibility of including the historical dataset in the calibration process, so that Θ_d is mapped through a Feed Forward Neural Network (FFNN) [14]. In this way, the model parameters depend directly on the set of market prices Q_d (or on its transformation, as done in Section 4.1), and a set of neural weights $\boldsymbol{\omega}$, i.e., $\Theta_d = f(Q_d, \boldsymbol{\omega})$.[1] It is possible to consider the calibration problem in terms of Supervised Learning, with the objective function being the minimization, over the weight space Ω, of the empirical mean of the cost functions over the historical dataset (the *error*):

$$\min_{\omega \in \Omega} \frac{1}{D} \sum_{d=1}^{D} \epsilon(\Theta_d, Q_d) = \min_{\omega \in \Omega} \frac{1}{D} \sum_{d=1}^{D} \epsilon(f(Q_d, \omega), Q_d). \tag{2}$$

Since the input space of prices is large, the input features of the FFNN are obtained by the Principal Component Analysis (PCA) [20] to reduce the dimensionality of the problem. The results of this process are shown in Section 4. Finally, as said in [16], the financial calibration process can have more than one local optima; for this reason, we implemented a two-fold calibration process: the first one is a global search for a candidate neighbor of a global optimum, while the second one is a local, gradient-based optimization process, as described in Section 5.

2.2 Offline vs Online

As mentioned in the introduction, with our approach we can learn the mapping $Q \longrightarrow \Theta$ by using the historical dataset (in our simulation, the training set), so that, as soon as new market data is available, the related parameter can be predicted on the fly. This approach is intended as *offline* learning. However, this might entail two issues: the first is related to the non-stationarity of the market; the similarity between market quotes decreases as time passes. Consequently, the goodness of the fit may deteriorate over time. The second issue is related to possible shocks, or changes in market behavior; in this case, the accuracy of the mapping may be lower, since market data can be placed in areas that have never been seen in the historical dataset. Hence, we can think of updating the calibrator: after having collected and evaluated a small batch of new samples (e.g., a weekly batch), this can be included in the training set so as to perform a local calibration with the overall, increased dataset. If the predicted parameters are close enough to the optimal ones, this latter procedure should be fast enough and can avoid the possible drawback mentioned before. This can be seen as an *online* setting, where learning dynamically adapts to new patterns in market quotes. In this way, the training set becomes larger and larger: on the one hand, it let us consider a larger part of the state space; on the other, computational complexity increases too. In order not to become unfeasible, the calibration can consider a stochastic sampling of the daily samples to be considered at each iteration.

[1]Dependence on exogenous market factors ξ is omitted here for simplicity.

3 PRICING MODEL: G2++

In this section, we define the parametric stochastic model that requires calibration. In particular, we specify the dynamics of the instantaneous, continuously compounded, annualized interest rate (the "short rate") as a two-additive factor Gaussian model (G2++). The G2++ is equivalent to the popular two-factor Hull-White interest rate model [7], which allows an easy-to-follow tractability of pricing formulas; in addition, it reproduces market prices of many vanilla swaptions with sufficient accuracy, thanks to its richer distributional properties compared to simpler models, such as Vasicek.

The G2++ model is defined by the following Stochastic Differential Equation:

Definition 3.1. Let $\mathbb{Q}$ be the Risk Neutral probability. A stochastic process defined on a filtered probability space $(\Omega, \mathscr{F}, (\mathscr{F}_t)_t, \mathbb{Q})$ is said to be a $G2++$ process if:

$$r(t) = x(t) + y(t) + \varphi(t), \tag{3}$$

where $\varphi \colon \mathbb{R} \longrightarrow \mathbb{R}$ is a deterministic function such that $\varphi(0) = r_0$; the processes $\{x(t) : t \geq 0\}$ and $\{y(t) : t \geq 0\}$ satisfy

$$dx(t) = -ax(t)dt + \sigma dW_x(t) \qquad x(0) = 0 \tag{4}$$

$$dy(t) = -by(t)dt + \eta dW_y(t) \qquad y(0) = 0, \tag{5}$$

with $\sigma \geq 0, \eta \geq 0$, and (W_x, W_y) being a two-dimensional $\mathbb{Q}$-Brownian Motion (BM) with instantaneous correlation $\rho \in [-1, 1]$:

$$dW_x(t)dW_y(t) = \rho dt. \tag{6}$$

The parameters to be calibrated $\Theta = \{a, b, \sigma, \eta, \rho\}$ are five: a and b are called Mean Reversion Speeds (MRSs), σ and η are the volatilities (VOLs) and ρ is the correlation between BM components. This model presents two sources of randomness: one is usually meant to drive short-term dynamics, the other is more important in the long term.

In this paper, the deterministic shift $\varphi(t)$ is considered as known, but twofold; indeed, we are considering a multi-curve framework, adopted in the standard market practice [4, 6], which properly takes into account two different curves to forecast and discount cash flows. Hence, through a daily bootstrap procedure from market curves

(Overnight Indexed Swaps for discounting, and EURIBOR 6m for forwarding), we can build two different shift curves: $\varphi^d(t)$ for the discounting case, $\varphi^F(t)$ for the forwarding case. Actually, what is used is their related terms $\Phi(t,T) = e^{-\int_t^T \varphi(s)ds}$, split into Φ^d or Φ^F. To generate a realistic calibration, the financial instruments necessary to build the dataset must be liquid and easily tractable under the chosen model. The simplest and most liquid derivative having interest rates as underlying are the European I.R. Swaptions. In particular, we will consider ATM Payer Swaptions, whose tenor and maturity are defined in the contracts (hence, they are specified by τ), and the consequent cash flows of the (quarterly) payments in the fixed leg are denoted by the set Γ. The arbitrage-free pricing formula of I.R. Swaptions according to the G2++ model can be synthesized as in Equation 7, which is equal to Theorem 4.2.3 in [7], with a slight modification to take into account the shift term:

$$\hat{Q}_\Theta(\tau) = \int_{-\infty}^{\infty} f_X(x) \sum_{\gamma\in\Gamma} \psi_\gamma(x.\bar{y}(x))dx, \tag{7}$$

where $f_X(x)$ is a Gaussian distribution having known explicit mean μ_x and variance σ_x evaluated in x, ψ_j is an explicit function, and $\bar{y}(x)$ is the unique solution of an equation in the following form:

$$\sum_{\gamma\in\Gamma} c_\gamma e^{-\alpha_\gamma x - \beta_\gamma \bar{y}(x)} = 0, \tag{8}$$

where c_γ, α_γ, and β_γ are explicit coefficients. The first approximation to Equation (7) is obtained by discretizing the distribution by means of the Gauss-Hermite quadrature [28]: by taking a set of nodes $\{x_\kappa\}$ and related weights $\{\omega_\kappa\}$, the adopted pricing formula becomes:

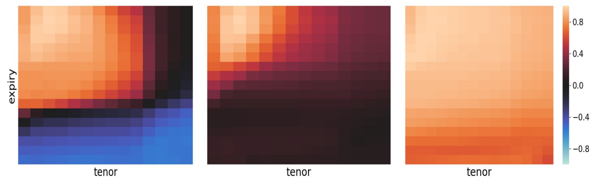
$$\hat{Q}_\Theta(\tau) \approx \frac{1}{\sqrt{\pi}} \sum_{\kappa=1}^{K} \omega_\kappa \sum_{\gamma\in\Gamma} \psi_\gamma\big(\sqrt{2}\sigma_x x_\kappa + \mu_x, \bar{y}(\sqrt{2}\sigma_x x_\kappa + \mu_x)\big). \tag{9}$$

The major problem related to the computation of these prices is that, for each day d in the training set, for each swaption considered (denoted with τ), and for each quadrature node x_κ, the solution of Equation (8) must be computed. This is the real bottleneck in the computational times of the calibration approach. In order to speed up the computation, the solution adopted consists, first of all, in the adoption of the Halley's algorithm, which in this case is easy to perform and achieves a cubic convergence [1], but above all in a massive parallelization by preforming pricing on GPU, where each block consists in a reference date, where the single swaptions are priced in each kernel.

4 DATASET AND PREPROCESSING

4.1 Single currency

First, we consider the single currency framework. The dataset is composed of Swaptions expressed in €, with a range of 17 possible expiries (from 1M to 30Y) and 14 possible tenors (1Y to 30Y), for a total of 238 combinations per day. Each daily sample (1420 samples, from 2013-28-06 to 2019-11-01) is provided with three 14x17 matrices: market prices, implied volatilities, and vega. However, there is no need to consider every single value (total of 714 per daily sample) as input for the neural network, because many features are highly correlated and there is the risk of adding unnecessary complexity



(a) prices **(b) Imp. volatilities** **(c) Vegas**

Figure 1: Correlation of the 2Mx3Y swaption with the other swaptions in the matrix (cell 2x3).

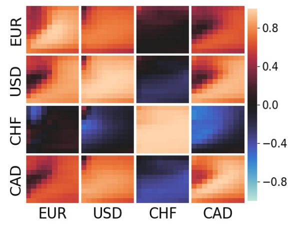
to the architecture. Indeed, it is possible to verify that the price with a specified tenor and expiry, corresponding to a specific value in the matrix, is highly correlated to the closest cells. This is due to the fact that close swaptions have similar tenors and expiries, and that the related cash flows exchanged between the counterparties are located at similar time intervals. While moving away from the considered swaption, the correlation decreases, assuming also strong negative correlations. This behavior is clearly depicted in Figure 1a, where the correlation matrix is the one related to the price of the swaption with 2 months as expiry and with a tenor of 3 years. As far as volatilities are concerned, it is possible to detect two groups of swaptions: the former collects swaptions with lower expiries and tenors (upper left corner in Figure 1b); the other swaptions are in the latter group. The swaptions in the same group are strongly correlated to each other; on the other hand, the two groups have a poor, negative correlation. Finally, the correlation matrix of the vegas presented in Figure 1c shows that all swaptions have generally high correlations.

As a consequence, the training samples are preprocessed with a dimensionality reduction procedure, namely the Principal Component Analysis (PCA [18]), an unsupervised learning technique that looks for a projection of the data onto a lower-dimensional linear space in such a way to minimize the loss of information, i.e., to maximize the variance of the projected data. The PCA process selected 12 components that account for over 99% of the explained variance, which is retained with over 75% in the first component, which leverages the differences of short-term and long-term swaptions in terms of prices and implied volatilities. The vega, instead, does not add information to the dataset. Hence, these new features are the ones included as input for the Neural Network. The same projection is applied in the test set, even if eventual changes in the underlying distribution of market data might result in a slow degradation of the efficiency of the process. We consider the possibility to apply PCA-based change detection techniques [22] for future works, in order to identify shifts in the distribution and to repeat the calibration accordingly.

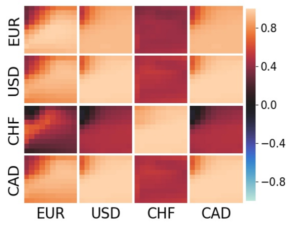
4.2 Multiple currencies

In a second instance, we consider a multi-currency framework, where we consider a dataset with four currencies, EUR, USD, CAD, and CHF, sharing the same mapping of the G2++ model parameters.[2] In order to have a homogeneous dataset, we only consider the common expiry/tenor pairs for the calibration, obtaining a 10x10

[2]The exogenous factors ξ, as the forward and discount curves, differ for each currency.



(a) prices



(b) Implied volatilities

Figure 2: Correlation matrices of the 3Yx6Y swaptions of four different currencies. Each row corresponds to the correlations of the 3x6 swaption in a specific currency with respect to the others. Each column specifies the currency of the dataset with which the swaptions are being compared.

swaption matrix per currency. The same issues described for the single-currency also apply in this case, but now we can also consider the cross-currency correlations. Since it is not possible to show the correlation matrix of the overall dataset, some particular features have been selected as an example. In particular, the swaption with a couple expiry/tenor of 3 and 6 years has been chosen for each currency. In Figure 1 it is possible to see the correlation of prices and volatilities of this swaption expressed in different currencies with the features of each dataset. By focusing on the diagonal matrices on the figure related to prices (Figure 2a), it is possible to verify what was already underlined: within the same currency, closer swaptions have higher correlations (3x6 swaption is placed in the middle of the matrices), except for *CHF*, where all swaptions are almost linearly correlated. The cross-correlation patterns suggest that the CAD and USD markets are highly correlated. Milder correlations hold when comparing these currencies with the *EUR* prices. It is noticeable that the correlation of the 3x6 EUR swaption price with all those in the *CHF* market is almost zero, while it becomes strongly negative when taking the *CAD* and *USD* swaptions into account. Figure 2b shows not only that the division of volatilities into two groups (short-term and long-term swaptions) still holds, but also that it is extended to a multi-currency framework. Again, the *CHF* market behaves differently. A similar consideration can be done for the vegas, with a generalized higher correlation between all swaptions. The PCA procedure has been also performed in this case with similar results.

5 ALGORITHM

In this section, we define the optimization algorithms adopted. As explained in [16], local optimization methods can result in an underperformance for calibration, since financial models are far from being convex and usually show multiple local minima. Hence, our calibration consists in the application of two sequential procedures: the first is a global, random-search method (Cross Entropy, CE) and, once a candidate neighborhood of the global optima is found, the well-known gradient-based algorithm, BFGS, is executed.

The CE method [26] is an optimization algorithm, based on Monte Carlo simulations. It is usually used to find the global minimum of noisy functions. The customized version of our algorithm simply consists of randomly sampling new parametrizations from a Gaussian distribution and the ones resulting in the best performance are selected; the empirical mean and variance are computed to update the Gaussian distribution for the next generation of samples. The algorithm stops when the variance of the best samples (across all the iterations) is below a certain threshold, which means that they are focused in a neighborhood of a candidate global minimum. This specific procedure, in the general idea, is similar to the sampling procedure followed in [16].

Once this global search ends, local calibration can be applied thanks to the well-known BFGS algorithm [8, 12, 27], a gradient-based algorithm in which an approximation of the inverse of the Hessian is used, as to include curvature information and accelerate the convergence.[3] Here it is possible to understand the main difference with the usual supervised learning techniques involving neural networks: in order to perform backpropagation, the main issue is that the loss function is not trivial since there is no direct supervision; in our case, we do not have a distance measure with respect to the optimal set of parameters, but a measure of the financial model error w.r.t. market data. However, it is still possible to compute the gradient, leveraging the independence of the calibrator from the interest rate model. This means that the model can be replaced without changing the general structure of the calibrator; furthermore, if the model is differentiable, it is possible to decompose the gradient of the *daily cost function* $\epsilon(\Theta_d, Q_d)$ defined in Equation 1, w.r.t. to a general network weight ω by means of the chain rule:

$$\frac{\partial \epsilon(\Theta_d, Q_d)}{\partial \omega} = \sum_{\theta \in \Theta_d} \frac{\partial \epsilon(\Theta_d, Q_d)}{\partial \theta} \frac{\partial \theta}{\partial \omega}, \tag{10}$$

where

$$\frac{\partial \epsilon(\Theta_d, Q_d)}{\partial \theta} = \frac{\sum_{\tau \in T_d} \alpha_\tau \left(\hat{Q}_{\Theta_d}(\tau) - Q_d(\tau)\right)^2 \partial_\theta \hat{Q}_{\Theta_d}(\tau)}{\epsilon(\Theta_d, Q_d)}. \tag{11}$$

In practice, in Equation (10) the structure of the calibrator is split into two components: the first is related only to the financial model adopted and consists, as expressed in Equation 11, in the computation of the Jacobian of the model prices w.r.t. the parameters chosen; the second component is related only to the network architecture, following the usual backpropagation, and is independent of the model $\mathcal{M}$ adopted.

As for the choice of the G2++ model we adopted, it is possible to compute all the derivatives of the predicted prices in Equation (7) w.r.t. the model parameters, with the same approximations adopted in Equation (9) and using the implicit function theorem to obtain the derivatives of the solution $\overline{y}(x)$. As a consequence, the computational complexity related to pricing is inherited for the gradient computation; consequently, this procedure is parallelized and performed on GPU, too.[4]

6 RELATED WORKS

In literature, there is a plethora of recent research regarding neural network applications on the prediction of market values, usually

[3]We are considering the full gradient computed over the whole training set; however, it is possible to consider also the stochastic gradient in the case of large datasets.
[4]Another possibility for computing the gradient, not implemented in this paper, can be the automatic differentiation that a tensor-based platform can provide.

in the form of implied volatilities, or their parametric representation [2, 5, 17, 19, 21]. In other cases, the goal is to find, using ANNs or CNNs, a proxy of a model, or some properties related to the samples generated by a learned model [17, 24, 29]. The pioneering work in financial inverse calibration is [15], which has some common points to our approach. Hernandez was the first one, up to our knowledge, to consider a neural network approach for financial calibration. However, the structure of the problem is set up as a standard supervised learning problem, since the dataset used for training contains as target variables the model parameters already obtained through external calibrations, such as the Levenberg-Marquardt local optimizer [25]; for this reason, we denote this approach as "indirect" calibration. These parameters are then fitted by the network through local optimization algorithms [15] or global methodologies [16] to avoid local minima. This framework is interesting, since it allows replacing the standard calibrator through transformations over the historical calibrated dataset: in this way it is possible to perform immediate evaluations over new market data. One of the major drawbacks of this approach is the fact that, in this way, the neural network is only used as a proxy for the external calibrator, reflecting its limitations: hence, added to the intrinsic model error, there is the external calibration error that cannot be neglected. In this paper, as shown in Section 5, we consider a "direct" calibration, consisting in the adoption of a global optimization, which is then followed by a local calibration that leverages the properties of the financial model chosen, which were ignored in the works mentioned above. Another drawback in [15] and [16], consist in the fact that it is not possible to consider an online calibration without starting the external optimization from the new samples, undermining some of the advantages of the work. In the framework of indirect calibration, other authors proposed similar applications on different models [10]. An interesting work in this context is [3], in which the optimization is twofold: since the major computation efforts in financial calibration lie in pricing (i.e., in the computation of $Q(\Theta, \tau, \xi)$), the authors propose to learn, in a first instance, a neural approximation of the pricing model; then, similarly to [15], another calibration is performed to learn the inverse mapping by using the learned proxy. This work is intriguing, but the proposed approach is still dependent on an external calibrator and its errors might be propagated and increased in the proxy, as in [23], in which a similar idea is proposed, by dynamically training different layers of a single neural network. Finally, as shown in [19], there is another major drawback in replacing the pricing model with its proxy consists in the possible violation of no-arbitrage conditions in the prices predicted from an ANN.

7 NUMERICAL SIMULATION

In this section, we provide the results of applying the calibration procedure. In the first subsection, we discuss the results obtained over the single currency calibration, with the dataset explained in Section 4.1. In a second instance, we consider a multi-currency scenario. In all cases, the neural network considered is composed of two hidden layers; the activation function adopted is the hyperbolic tangent. The coefficient K in Equation (9) is empirically set to 20, and the weight α_τ related to Equation (1) is set as the squared inverse vega of each swaption [13]. As for the details of the algorithms, CE collects 100 samples each time and computes the mean and variance for the best time for each iteration, plus the best 3 obtained over all iterations. CE stops when the sample variance is below 0.3, while BFGS stops when the gradient norm is below 10^{-6} or after 250 iterations.

Table 1: Train and validation error obtained by using different hidden neurons per hidden layer [mean ± std on 5 runs].

Neurons	Train error	Validation error
3	0.3799 ± 0.0181	0.5426 ± 0.0714
5	0.3526 ± 0.0012	**0.4907** ± 0.0382
8	0.3497 ± 0.0008	0.5290 ± 0.0262
10	0.3491 ± 0.0003	0.5606 ± 0.0531
20	0.3482 ± 0.0008	0.5822 ± 0.0430
30	0.3486 ± 0.0012	1.0970 ± 0.0793

The neural architecture (with different numbers of neurons) is built by considering the sigmoid for the hidden layers and the hyperbolic tangent for the output layer as activation function, with a different reshaping for every final parameter, in order to set a fixed range of possible values, such as [-1,1] for the correlation and positive values for the volatilities. The Mean Reversion Speeds are not constrained to positive values.

7.1 Validation

In order to optimize the number of neurons per hidden layer, we first performed a validation phase on the single currency scenario: by using 40% of the dataset as a training set, and the subsequent 20% as a validation set, we performed 5 different runs of the calibration process for each one of the considered possibilities. The overall feedback, with average values and standard deviations, is shown in Table 1. In particular, it is possible to see that, as expected, very small architectures are not sufficient to properly build the desired mapping; on the other side, from only 5 neurons per layer, the training error does not change too much, which means that the daily error is close to the model error; however, it is possible to see the overfitting effect in validation. With 30 or more neurons, calibration starts to become more difficult for CE: due to the higher dimensionality of the weight space, it becomes easier for the procedure to get stuck in local optima: in order to avoid this issue, the number of samples collected per iteration should be increased, resulting in very long computational times. Hence, we continued our experiments by using only 5 neurons.

After validation, we considered a calibration with a split in the dataset, where 60% is used as the training set. Another 20% builds the online set: as explained in Section 2.2, one new daily sample is considered iteratively; after the evaluation of the daily feedback, it is added to the training set and then BFGS is performed again, with a maximum number of iterations equal to 150.

The same validation procedure was performed for the multi-currency case. However, since each currency has the swaption market values expressed in its own currency, the region of the state space included in the dataset is larger; hence, as shown in Table 2, more hidden neurons are needed to represent the mapping. The issue related to the convergence to local minima with larger

Table 2: Train and validation error, multi-currency case [mean ± std on 5 runs, 2 hidden layers].

Neurons	Train error	Validation error
5	0.35299 ± 0.1497	0.41790 ± 0.1224
10	0.31133 ± 0.0585	0.38520 ± 0.0717
20	0.28495 ± 0.0286	**0.36751** ± 0.0523
30	0.33382 ± 0.0415	0.44688 ± 0.0761
40	0.50499 ± 0.0974	0.56397 ± 0.0910

networks here is amplified, thus having large train errors, especially with 40 neurons per layer. By looking at the results, we used 20 neurons per hidden layer to perform the final (and online) training.

7.2 Single currency

In this section, we deal with the calibration in the single currency setting shown in Section 4.1 and using 5 neurons per hidden layer. After training on 60% of the historical dataset, we first evaluated the feedback on the remaining part of the dataset, then simulated the *online* calibration: we iteratively considered 5 the next new daily samples and added them to the training set, thus updating the weights by performing only the local calibration with a maximum of 50 gradient steps. The online procedure is stopped when the overall training set contains 80% of the entire dataset, while the remainder is used for testing purposes.

The results in Figure 3 show the curves of the five model parameters and the related daily feedback. In particular,the Mean Reversion Speeds of the two processes defining G2++ model (a and b) have a different order of magnitude. On the other hand, the volatilities (σ, η) are in a similar range of values. At last, there is always a strong negative correlation between the Brownian Motions, especially since 2015. As a consequence, it is possible to interpret that the processes $x(t)$ and $y(t)$ have the same amplitude of random noise, with opposite directions and a different mean reverting reaction: one component has a stronger force that drives the dynamics back to its mean, while the other one is milder as if one process is driving the short-term dynamics, in contrast to the long-term process. It is possible to see that the patterns of the MRSs and their related volatilities are similar: indeed a and σ show a correlation of 69% and η have a correlation of 70% with b, which shows different behaviors through time.

Feedback and online adaptation. By looking at the results of the offline calibration, some considerations can be made with regards to the daily feedback, shown in the last plot in Figure 3: even if the gain w.r.t. a simpler model like Vasicek is clear (the results are obtained through Levenberg-Marquardt algorithm [25]), we can see that there is a degradation in time of the out-of-sample performance. However, by considering the online calibration, it is possible to reduce this effect and get a better fit on out-of-sample data. This is mainly due to the auto-correlation between the series of market values: the market trend slowly moves to unseen regions of the state space and online calibration can improve the fit in those areas. However, there is still a loss of information obtained from PCA: this transformation is performed by considering the most useful pieces of information of the training set, hence it might lose some relevant features in the new samples, nor it is possible to simply change the transformation without restarting the calibration. In order to better understand the effects of time and trends in market behavior with online learning, we performed the same training process by considering a new dataset, in which the daily samples are shuffled (hence there is no day-to-day correlation). In this case, there is only a small gain between offline and online calibration, due to the sole inclusion in the dataset of more samples. This gain is negligible when compared to the one obtained with the actual online calibration on the ordered dataset, as shown in Figure 4.

Table 3: Train and Test error obtained by using direct calibration (ours) and indirect, once the best model parameters are given from an "oracle" [mean ± std on 5 runs].

Method	Neurons per layer	Train error	Test error
Oracle		0.3629	0.4272
Indirect	5	0.5017 ± 0.0380	0.8172 ± 0.0663
Indirect	20	0.4334 ± 0.0362	0.4953 ± 0.0280
Indirect	100	0.4226 ± 0.0203	0.5771 ± 0.0513
Direct	5	0.3699 ± 0.0026	0.5064 ± 0.0140

Computational times. As specified in Section 1, the main advantage of such calibration consists in the time needed to return a new set of parameters once a set of market data is provided. Indeed, the day-by-day calibration, performed with standard techniques (L-BFGS-B [9]), takes on average 19 seconds per daily set of market data. Instead, in our case the evaluation simply consists in providing the output of a trained neural network, which is several order of magnitude faster since, on average, it takes $1.2 \cdot 10^{-3}$ seconds. This is due to the fact that the training time is performed offline, thus not affecting the real-time operations in the desks. The overall computational time for training the network amounts to 190 minutes, which is in any case faster than the total time needed to perform the daily calibration on the training set (for a total of 285 minutes).

Direct and indirect calibration: comparison. In Table 3 we compare our direct approach with the indirect calibration, where the cost function is measured as quadratic loss w.r.t. the best parameters for each daily sample (*oracle*). In general, direct calibration performs better, especially when considering the same network architecture (although, indirect calibration with 20 neurons for both the hidden layers obtains better test results with no statistical evidence). This is due to the fact that the goal of indirect calibration is to obtain outputs which are closer to the given target: however, since the pricing function is not regular, this might still lead to achieve feedbacks which are far from the optimum.

7.3 Multi currency

In the multi-currency scenario introduced in Section 4.2, the dataset is made of approximately 945 business days per currency, from May 2015 to Jan 2019. The calibration process is the same as in the single currency, but this time, as reported in Table 2, the chosen network is built with 20 hidden neurons per hidden layer, since the space to learn the mapping is broader. There are some considerations

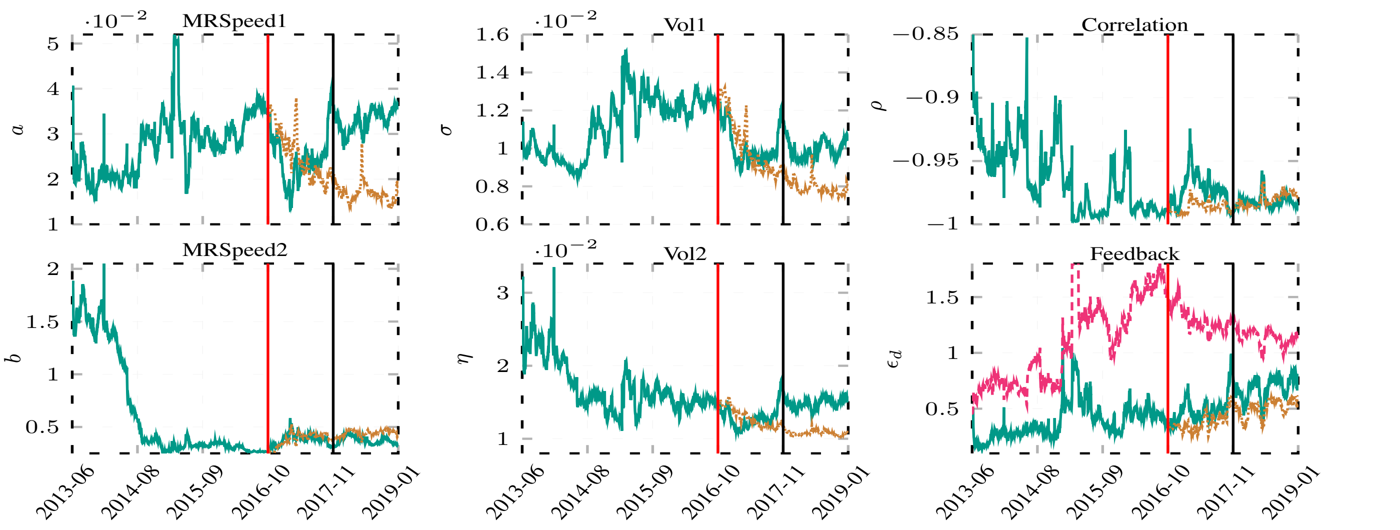


Figure 3: Calibration on the € dataset. The first five plots show the behavior of the G2++ parameters, while the last one illustrates the daily feedback. The dense teal curve represents the offline calibration (trained up to the red line). The dotted brown line denotes the online calibration, starting from the red vertical line up to the black line (and then tested) with a batch size of 5 daily samples. The magenta dashed line represents the daily error obtained from the calibration of the Vasicek model.

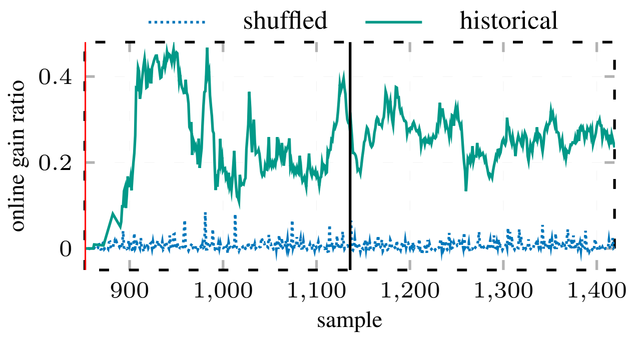


Figure 4: Daily Feedback relative gain obtained with online calibration: comparison between using the dataset with chronological order (historical) and shuffled daily samples. The vertical black line denotes the end of the online phase.

to make upon this setting: first of all, the subset of swaptions T_d considered is composed of the most liquid ones; hence, the daily feedback curve for EUR depicted in Figure 5is lower than the single-currency one, because the RMSE is computed from a smaller set. As a consequence, the excluded swaptions are the least liquid ones, with higher uncertainties in market behavior, hence the noise in the measures is reduced and the same holds for the degradation in the test set. In Figure 6 it is possible to see that the parameters detected for the different currencies behave in a similar way and suggest similar dynamic properties. However, there are some differences in the EUR curve w.r.t. the single-currency framework: the most important one is the presence of a negative Mean Reversion Speed, which, in the previous case, was not detected.

8 CONCLUSIONS AND FUTURE WORKS

Improving calibration speed of stochastic interest rate models can be valuable for investment banks and more generally for agents in the derivatives market. In this work, we propose a black-box calibration of a two-factor interest rate model (G2++) in a multi-curve setting, consisting of a FeedForward Neural Network which is independent of external calibrators, as it learns directly from market data by means of the financial model (differently from state of the art NN-based calibration methods). The combination of a global (stochastic) and a local (gradient-based) optimization algorithm is proposed, along with the parallel implementation on GPU, which provides a significant speed up. Training can be performed offline by using the historical dataset, while small improvements can be obtained online. Finally, the approach is evaluated in two scenarios, which consider a single-currency, and a multi-currency framework.

In future works, we will consider the possibility to apply change detection techniques [22] to trigger online updates or to restart PCA projection. Moreover, the calibration task in the multi-currency setting will be extended to include a consistent specification for foreign exchange rate dynamics, in order to achieve fast calibration of a fully specified cross-currency derivatives pricing model.

ACKNOWLEDGEMENTS

The research was conducted under a cooperative agreement between ISI Foundation, Intesa Sanpaolo IMI Corporate & Investment Banking Division and Intesa Sanpaolo Innovation Center.

REFERENCES

[1] G Alefeld. 1981. On the convergence of Halley's Method. *The American Mathematical Monthly* 88, 7 (1981), 530–536.
[2] Panayiotis C Andreou, Chris Charalambous, and Spiros H Martzoukos. 2010. Generalized parameter functions for option pricing. *Journal of banking & finance* 34, 3 (2010), 633–646.

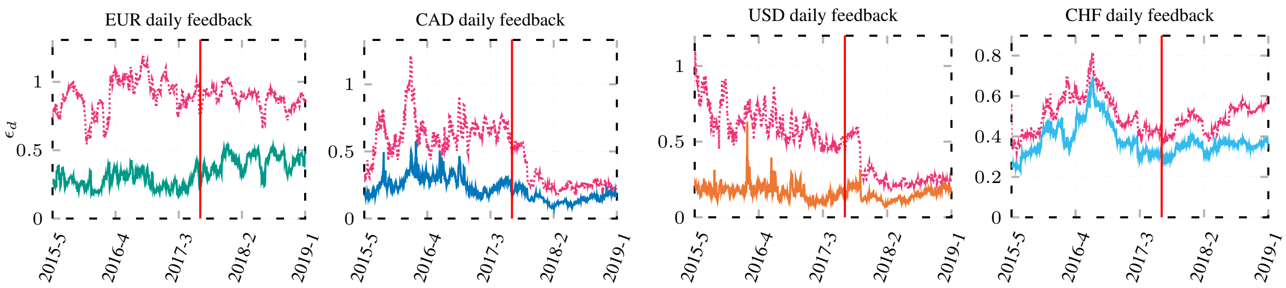


Figure 5: Multi-currency offline calibration: comparison of G2++ offline daily error (solid line) with Vasicek daily error (magenta dotted line). The red vertical line defines the train/test split.

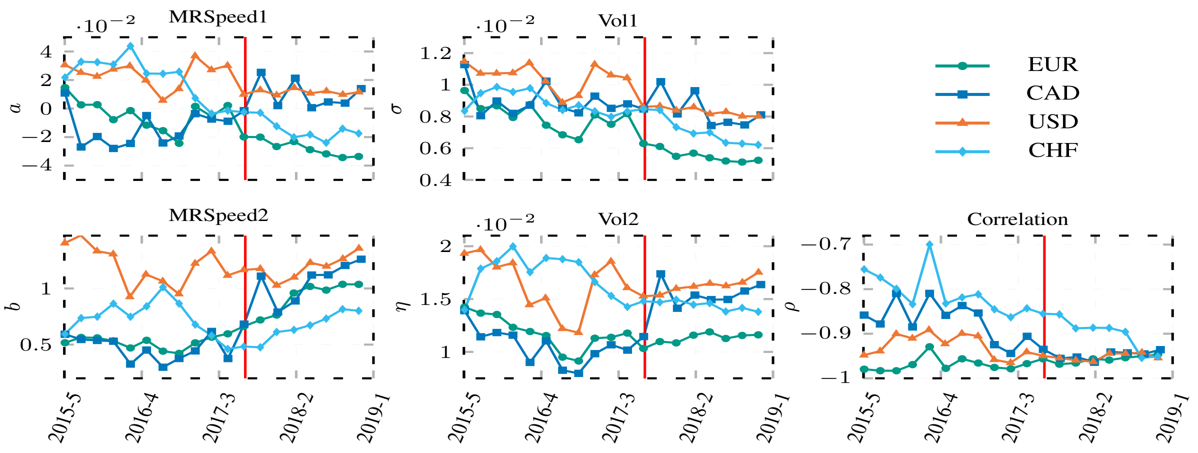


Figure 6: Multi-currency offline calibration: patterns of G2++ parameters (one point every 30 calibrated dates).

[3] Christian Bayer, Blanka Horvath, Aitor Muguruza, Benjamin Stemper, and Mehdi Tomas. 2019. On deep calibration of (rough) stochastic volatility models. *arXiv preprint arXiv:1908.08806* (2019).

[4] Marco Bianchetti. 2008. *Two Curves, One Price :Pricing & Hedging Interest Rate Derivatives Decoupling Forwarding and Discounting Yield Curves.* MPRA Paper 22022. University Library of Munich, Germany. https://ideas.repec.org/p/pra/mprapa/22022.html

[5] Daniel Alexandre Bloch. 2019. Neural Networks Based Dynamic Implied Volatility Surface. *Available at SSRN 3492662* (2019).

[6] D. Brigo and F. Mercurio. 1998. On Deterministic Shift Extensions of Short-Rate Models. Internal Report, Banca IMI, Milan.

[7] D. Brigo and F. Mercurio. 2001. *Interest Rate Models - Theory and Practice.* Springer-Verlag.

[8] Charles George Broyden. 1970. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics* 6, 1 (1970), 76–90.

[9] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing* 16, 5 (1995), 1190–1208.

[10] Georgi Dimitroff, Dirk Roeder, and Christian P Fries. 2018. Volatility model calibration with convolutional neural networks. *Available at SSRN 3252432* (2018).

[11] Bank for Internation Settlements. 2020. *OTC derivatives outstanding statistics.* https://www.bis.org/statistics/derstats.htm

[12] Donald Goldfarb. 1970. A family of variable-metric methods derived by variational means. *Mathematics of computation* 24, 109 (1970), 23–26.

[13] Wolfgang Karl Härdle, Cathy Yi-Hsuan Chen, and Ludger Overbeck. 2017. *Applied quantitative finance.* Springer.

[14] Simon S. Haykin. 2009. *Neural networks and learning machines* (third ed.). Pearson Education, Upper Saddle River, NJ.

[15] Andres Hernandez. 2016. Model calibration with neural networks. *Available at SSRN 2812140* (2016).

[16] Andres Hernandez. 2017. Model Calibration: Global Optimizer vs. Neural Network. *Neural Network (July 3, 2017)* (2017).

[17] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. 2019. Deep learning volatility. *Available at SSRN 3322085* (2019).

[18] H. Hotelling. 1933. *Analysis of a complex of statistical variables into principal components.* Journal of Educational Psychology.

[19] Andrey Itkin. 2019. Deep learning calibration of option pricing models: some pitfalls and solutions. *arXiv preprint arXiv:1906.03507* (2019).

[20] Ian T Jolliffe. 1986. Principal components in regression analysis. In *Principal component analysis.* Springer, 129–155.

[21] Joerg Kienitz, Sarp Kaya Acar, Qian Liang, and Nikolai Nowaczyk. 2019. Deep Option Pricing-Term Structure Models. *Available at SSRN 3498398* (2019).

[22] Ludmila I Kuncheva and William J Faithfull. 2013. PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems* 25, 1 (2013), 69–80.

[23] Shuaiqiang Liu, Anastasia Borovykh, Lech A Grzelak, and Cornelis W Oosterlee. 2019. A neural network-based framework for financial model calibration. *Journal of Mathematics in Industry* 9, 1 (2019), 9.

[24] William A McGhee. 2018. An artificial neural network representation of the SABR stochastic volatility model. *Available at SSRN 3288882* (2018).

[25] Jorge J Moré. 1978. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis.* Springer, 105–116.

[26] Reuven Y Rubinstein and Dirk P Kroese. 2013. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning.* Springer Science & Business Media.

[27] David F Shanno. 1970. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation* 24, 111 (1970), 647–656.

[28] NM Steen, GD Byrne, and EM Gelbard. 1969. Gaussian Quadratures for the Integrals $\int_0^\infty e^{-x^2} f(x)dx$ and $\int_0^b e^{-x^2} f(x)dx$. *Math. Comp.* (1969), 661–671.

[29] Henry Stone. 2019. Calibrating rough volatility models: a convolutional neural network approach. *Quantitative Finance* (2019), 1–14.