Model Calibration: Global Optimizer vs Neural Network

Andres Hernandez
PwC
andres.hernandez@gmx.net

July 3, 2017

Abstract

Calibration of financial models can have more than one local minima present, requiring the use of global optimization techniques to properly calibrate them. In general, calibrating with a global optimizer will be a slow operation. An artificial neural network, properly trained, can replicate the same behaviour, providing the calibrated parameters in a fraction of the time. In the following, a simulated annealing global optimizer is used to calibrate a two-factor Hull-White model. A neural network is then trained to replicate it, and back-tested out-of-sample with good results.

1 Introduction

The use of pricing models in finance requires the calibration of model parameters to observed market data. For models that cannot replicate all the market data by construction, the process applied is generally an optimization method that attempts to minimize a chosen cost function that compares market prices with model implied prices.

The most widely used calibration techniques use local optimization methods that progressively work towards a better solution and would declare success upon being unable to improve significantly on the current solution. However, the models and cost functions used in finance were not chosen for producing convex problems, with a guaranteed single minimum. Rather, the models and cost functions were primarily chosen for displaying behaviour which was deemed to approximate important aspects of financial markets.

In reality, even simple models like the Heston stochastic volatility model can display a multitude of local minima in certain parameter regions [2]. Using a local optimization method can result in underperformance in this cases. To avoid exhausting the search budget with the first minimum found, there are alternative global optimization methods.

Electronic copy available at: https://ssrn.com/abstract=2996930

Without specific knowledge of the calibration function, normally not available for models used in finance, a global optimization method will be unable to guarantee that the true global minimum will be found. Instead, the procedure attempts to balance two competing objectives: finding the local minimum versus exploring new regions. The nature of the problem forces the method to not always move towards a better solution. This results in a more expensive proposition than a local optimizer, generally being noticeable slower.

Artificial neural networks offer an opportunity to sidestep the cost of using a global optimizer. In [3], it was shown how neural networks can be used for this precise purpose. In there, a neural network was used to replicate the calibration of a (one-factor) Hull-White model with a local optimizer. In the present work, the calibration of a Hull-White two-factor model with a global optimizer will be reproduced via a neural network, allowing for the calibration of such a model in a matter of milliseconds in comparison with a simulated annealing global optimizer potentially taking a few minutes.

This work is structured as follows. Section 2 lays out the calibration problem that will be performed. Section 3 details the neural networks that is used to replace the global optimizer. Section **??** provides the comparison of the simulated annealing global optimizer, the neural network, and a k-nearest neighbor method, as another machine learning approach. Section 4 provides some closing comments and perspectives for future work.

2 Calibrating a two-factor Hull-White model

The two-factor Hull-White model [4] contains an extra source of randomness, and provides a more realistic dynamic of the term structure than the original one-factor Hull-White model. It takes the following form

$$\begin{aligned} dr(t) &= (\omega(t) + u(t) - \alpha r(t))\,dt + \sigma_1 dW_1(t) \\ du(t) &= -bu(t)dt + \sigma_2 dW_2(t) \end{aligned} \tag{1}$$

with $dW_1(t)dW_2(t) = \rho dt$. The four parameters α, σ_1, σ_2, and b are positive, and shared across all option maturities. $\omega(t)$ is picked to replicate the current yield curve.

The model implementation, which was actually used is the equivalent two-additive-factor gaussian model[19] implemented in QuantLib [1].

The calibration problem consists in finding parameters $\theta = (\alpha, \sigma_1, \sigma_2, b, \rho)$, which minimize a cost function

$$\theta = \underset{\theta^* \in S \subseteq \mathbb{R}^5}{\text{argmin}}\ \text{Cost}\left(\theta^*, \{Q^{mkt}\}; \{\tau\}, \phi\right) = \boldsymbol{\Theta}\left(\{Q^{mkt}\}; \{\tau\}, \phi\right). \tag{2}$$

The calibration problem can be seen as a function with N arguments and

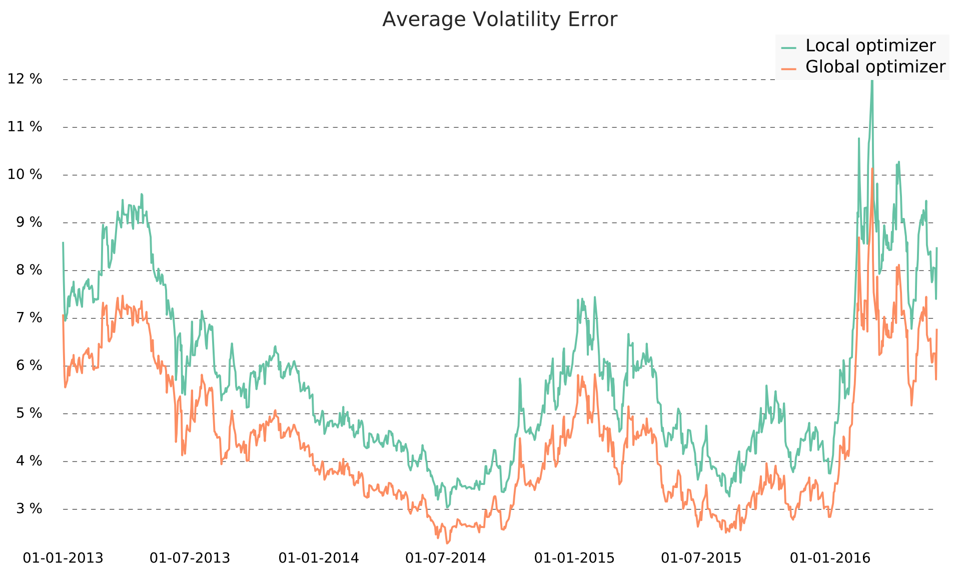

Figure 1: Comparison of local optimizer against global optimizer

$n = 5$ outputs

$$\mathbf{\Theta} : \mathbb{R}^N \mapsto S \in \mathbb{R}^n. \tag{3}$$

In the current exercise, N will have a value of 200. It is this function, $\mathbf{\Theta}$, which will be approximated by a neural network in the next section.

The model will be calibrated for the collected historical data, which comprises log-normal ATM volatility quotes for GBP from January 2nd, 2013 to June 1st, 2016. The option maturities are 1 to 10 years, plus 15 and 20 years. The swap terms from 1 to 10 years, plus 15, 20, and 25 years. Giving a total of 156 swaptions to be used equally weighted as calibration instruments.

For the yield curve, the 6M tenor Libor curve was used: bootstrapped using a monotonic cubic spline interpolation of the zero curve, and built on top of the OIS curve. Only FRAs and swaps were used to bootstrap the curve. When serving as input to the neural network, the yield curve is discretely sampled on 44 points: 0, 1, 2, 7, 14 days; 1 to 24 months; 3 to 10 years; plus 12, 15, 20, 25, 30, 40, and 50 years.

For each of the historical examples, the model is calibrated using first a Levenberg-Marquardt local optimizer, and later using a simulated annealing global optimizer. The global optimizer consistently outperformed the local optimizer, as can be seen in Fig.(1).

On average, the global optimizer is providing around a 20% improvement over the local optimizer in terms of the average error. However, in absolute terms it is a difference of only about 1%. Whether that difference presents a problem or not, is of course dependent on the particular application.

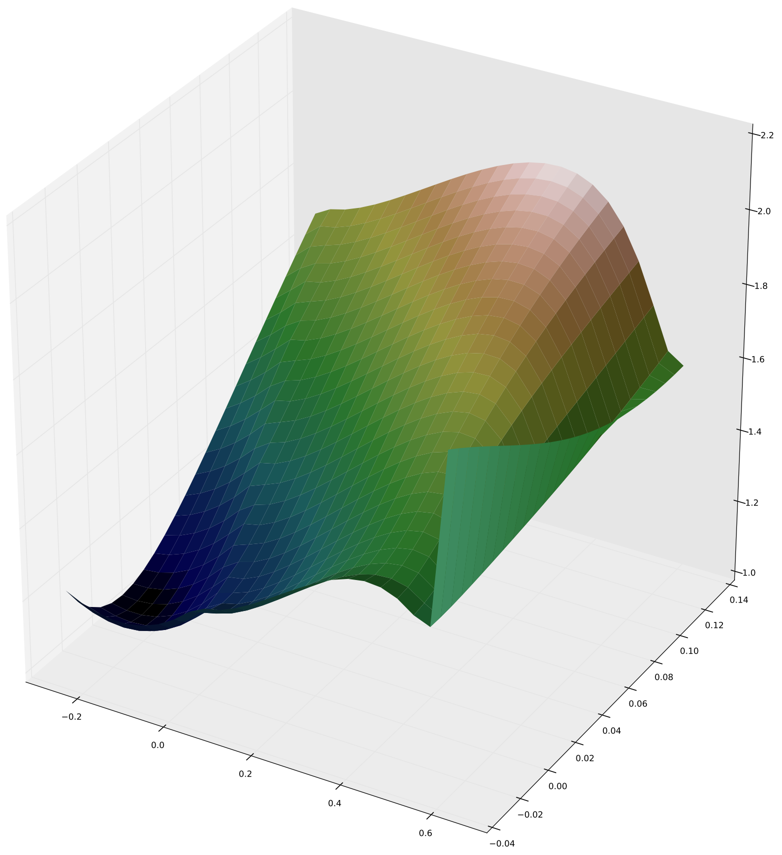

Figure 2: Comparison of local optimizer against global optimizer

The reason for the difference can be appreciated in Fig. (2), which displays a slize of the optimization as of the 24th of March, 2013. As the parameter space is five dimensional, it is not so straightforward to show that an optimizer could fall into a local minimum. In Fig. (2), a plane is shown, which joins the point found by the global optimizer (lower left-hand corner), the point found by the local optimizer (lower right-hand corner), and the starting point (upper right-hand corner). The figure suggest that if the starting point is found to the right of the hump, or even at the very top of the hump, the local optimizer will roll down the steep fall and end up in the local optimum. Technically, the optimization method is not restricted to this plane (or another, perhaps the one that joins the global optimum, the starting point and its gradient). However, Fig.(1) has already shown that there is indeed a difference.

3 Calibration with Neural Networks

In [13] neural networks were used to calibrate a one-factor Hull-White model. Neural networks have been used for model calibration in other contexts as well, see e.g. [14], [15], [16], [17], and [18].

To perform the calibration process via a supervised training method, e.g. neural network, proceeds by splitting up the task into two parts: first, training the machine learning model on a set of training examples; and second, extracting predictions from the learned model based on new input.

The advantage of such a setup lies in the speed with which the calibrated model parameters can be provided when needed. The expensive part of the calculation is effectively shifted to an off-line mode, i.e. offloaded to a uncritical time. For example, if needed the supervised model could be retrained over the weekend or overnight, outside of normal use.

In general, the collection of a large enough set of training examples is a hurdle to use neural networks. In contrast to model calibration in some other contexts, the case of calibrating financial models presents the difficulty of not modelling a physical system, which can be directly sampled as often as required. It is the acquisition of a training set big enough for the current purposes, that constitutes the largest effort in this exercise.

The collection of the training set proceeds in the way described in [13]:

- Obtain errors for each calibration instrument for each day
- Take the natural logarithm of the calibrated parameters, except for rho, which can be negative
- Rescale parameters, yield curves, and errors to have zero mean and unit variance

- Apply dimensional reduction, e.g. via PCA keeping enough eigencomponents for a given explained variance of 99.5%
- Calculate covariance of reduced factors
- Generate random, normally distributed samples consistent with given covariance
- Revert all transformations: rescale to original mean, variance, and dimensionality, and take exponential of firt four parameters
- For each sample, select a reference date randomly from the set used for covariance estimation
- Obtain implied volatility for all calibration instruments, adjusting them by the drawn error

The first step, obtaining errors for all calibration instruments, is of vital importance, as otherwise the neural network will likely not be able to reproduce the calibration function $\boldsymbol{\Theta}$ properly. If the financial model is able to perfectly reproduce the market prices, or if the errors were evenly distributed around the market price, this step would not be needed. However, in the current case, where certain regions of the volatility surface will tend to have errors in a particular direction, then without adjusting the theoretical prices appropriately, the neural network would in fact be learning the wrong $\boldsymbol{\Theta}$, and will result in a poor performance.

The next few steps are basically rescaling and adjusting the parameters with standard data science techniques, which generally improve the predictive power of a model.

Alternatively, the model could be trained with the original dimensionally-reduced samples. This was not investigated, as the current step resulted in a good replication of $\boldsymbol{\Theta}$.

Once the training set is available, a suitable neural network configuration is needed, together with an appropriate training method. Different models will best be proxied by different neural networks, with the number neurons, layers, and connections between them changing from application to application. Determining such a suitable configuration involves a so-called hyperparemter optimization, whereby the different parameters determinig a network, e.g. again the number of neurons per layer, are varied and an optimal design is chosen.

The process of hyper-parameter optimization is a recurring problem in neural networks, and there are standard approaches that can give good results. The simplest examples are grid search and manual search. In the former a multi-dimensional grid of parameters is prepared and the configuration with the best test results is picked. There exist, of course, more developed approaches, e.g. random search [?] or more formal search strategies [5].

In the current exercise, a mixture of grid and manual search was used due to resources limitations. As each set of hyper-parameters requires that a new neural network be trained, a grid search can spend significant amount of time pursuing avenues which are unlikely to bear fruit. After a limited grid search, a manual inspection selected the areas on which a further grid search would concentrate. This was repeated in an iterative process.

The resulting configuration is detailed in Fig.(3). It has 9 hidden layers, each with 64 residual learning blocks [20]. The residual was taken every layer, instead of the standard two layers, as the configuration showed better performance. Each layer consists of a batch normalization part [21], an ELU[6] activation function, a fully connected weight layer, a dropout section active during training [22], and lastly the residual part, recombining the output of the layer with the previous input.

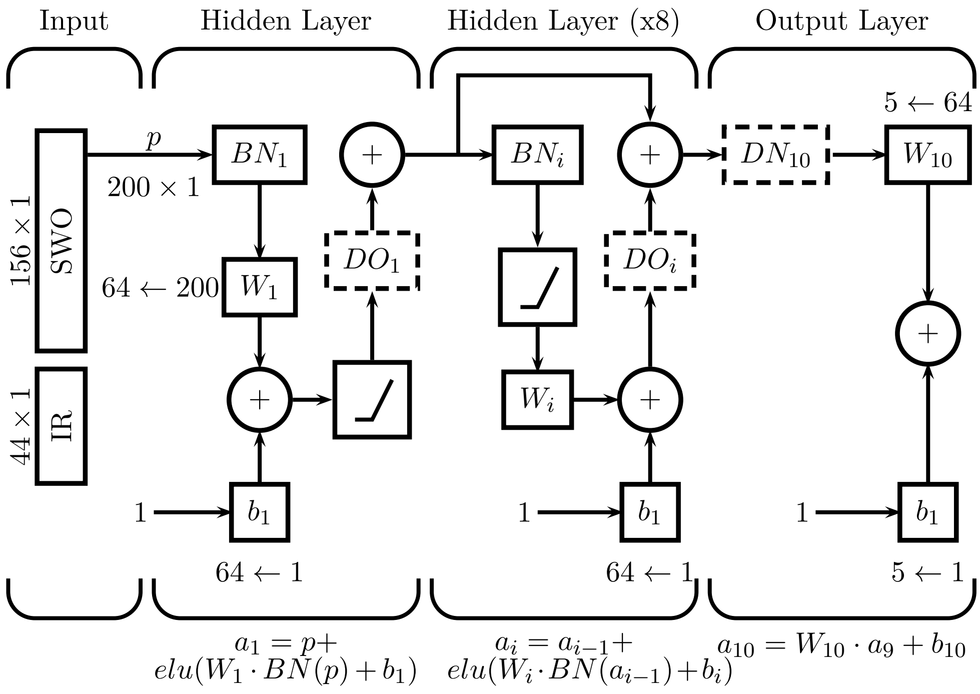

Figure 3: Feed-forward neural network used for two-factor Hull-White calibration

The model has over 46 thousand parameters. The dropout units are activated during training, randomly dropping a fraction r of the incoming input at each evaluation, preventing overfitting and producing more robust learning. The network in fact acts as an ensemble method, averaging the result of several effective models. The optimal dropout rate was found to

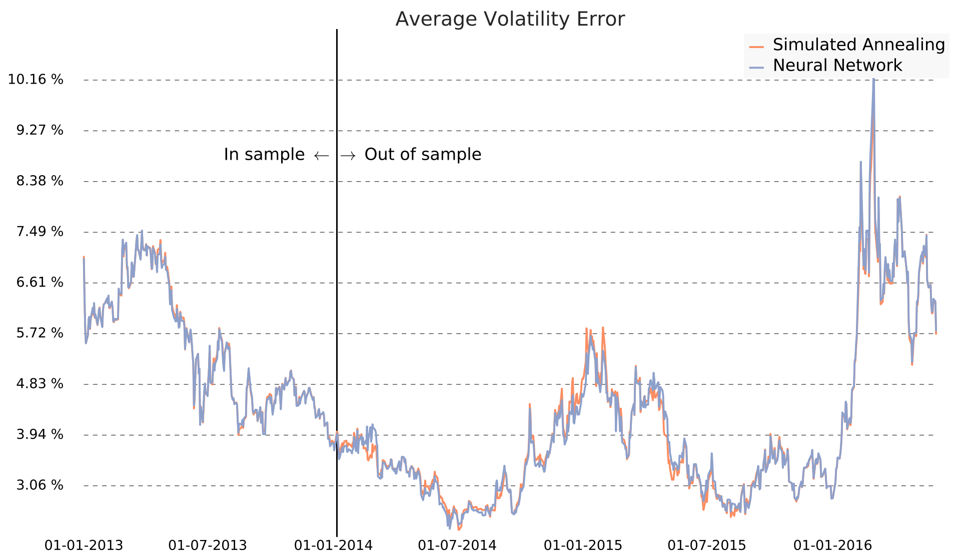

Figure 4: Comparison of global optimizer vs neural network

be 1/5 and was the same for all layers. A few experiments tried to use a different rate for the last layer, but were suboptimal.

As was seen in [13], the trained network will loose predictive power after some time, therefore the network will be retrained every 2 months. For each training procedure, a rolling 1-year window is used to calculate the covariance used in the creation of the training set. The result is shown in Fig.(5). Starting from January 2014, the prediction is made out of sample.

4 Conclusions

The use of pricing models in quantitative finance necessitate their calibration to relevant market data in an optimization process which can be time consuming. While the choice of model is informed by many considerations, the time required to perform the calibration is certainly one of them. In certain settings, like trading desks, calibration time can certainly be important.

We propose a novel approach for calibration, which can offer significant time savings during the actual calibration over the traditional optimization based approach. The improvement in performance is achieved by utilizing neural networks to approximate the calibration problem, off-loading the bulk of the calculations to a training phase, which can be done offline and at a convenient time.

As an example, the approach was tried out with the Hull-White model calibrated to 156 swaptions. The results are shown in Fig.(**??**-**??**). As can be seen, the out-of-sample backtesting shows good behaviour for between 6 months and 1 year after the end of the period used in the correlation esti-

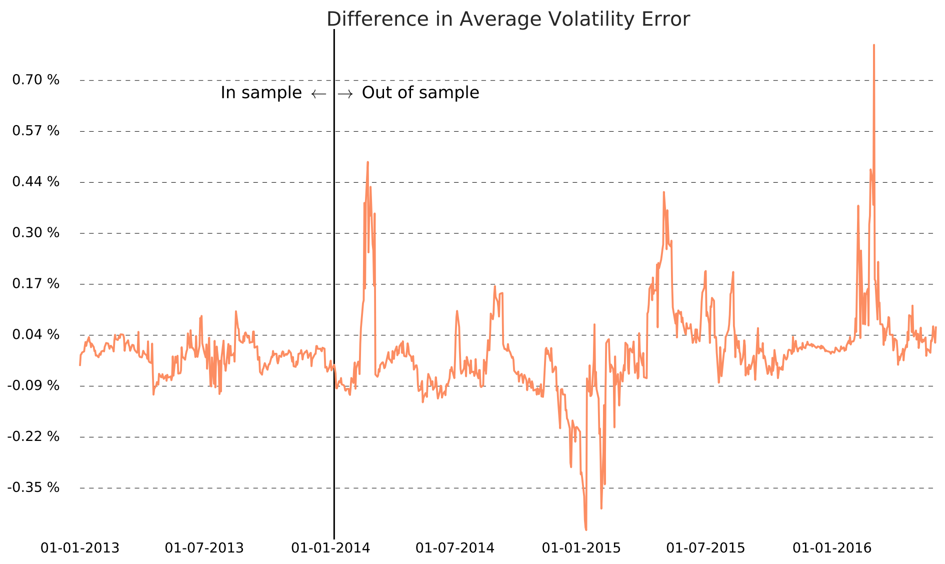

Figure 5: Difference between global optimizer and neural network

mation. After that, a significant degradation of the calibration performance of the neural network was observed.

The performance degradation beyond 6 months is not problematic, as one could simply retrain the model periodically, and remain within acceptable bounds. Also, in critical environments, safeguards could be set up to monitor the reliability of the solution. For example, a periodic parallel calibration using the traditional approach could be used to ascertain the reliability of the neural network vis-a-vis the current market state.

Overall, it can be seen how the methodology described in this work presents a substantial time improvement. Moreover, the time required to calibrate the model upon being presented with new inputs is quite uniform, as shown by the very low standard deviation for the example presented. This is a direct consequence of Fig. (3). While the training might be time consuming, the actual calibration involves multiplying only a few matrices and evaluating a handful of exponentials. The speed improvement will directly carry over to more complicated models.

An example with a more complicated calibration procedure, requiring the use of a global optimizer, will be produced shortly in a separate work. Further points of study involve using a parametrized correlation. Such a possibility would allow for training with states that not only reflect the current market environment, but also possible future developments, which could extend the life-span of a trained model. It is important to mention at this point that unlike in Monte-Carlo calculations, it is not necessary for the sampling set to correctly be able to predict how the future market conditions will statistically look like, but rather simply to contain samples in sufficient quantities that mimic such possible conditions.

References

[1] QuantLib A free/open-source library for quantitative finance, `http://www.quantlib.org`

[2] Yavor Kovachev, *Calibration of stochastic volatility models*, U.U.D.M. Project Report 2014:24

[3] Andres Hernandez, *Calibration with neural networks*, Risk Magazine, June 2017

[4] John Hull and Alan White, *Numerical Procedures for Implementing Term Structure Models II: Two-Factor Models*, Journal of Derivatives, Winter 1994

[5] Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams, *Practical Bayesian Optimization of Machine Learning Algorithms*, Neural Information Processing Systems, 2012

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*, arXiv:1511.07289 [cs.LG]

[7] Timothy Dozat, *Incorporating Nesterov Momentum into Adam*, `http://cs229.stanford.edu/proj2015/054_report.pdf`

[8] G. Cybenko, *Approximations by superpositions of sigmoidal functions*, Mathematics of Control, Signals, and Systems, 1989. Volume 2, Issue 4, pp. 303-314

[9] Kurt Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, Volume 4 Issue 2: 251-257, 1991

[10] R. J. Frank, N. Davey, S.P. Hunt, *Time Series Prediction and Neural Networks*, Journal of Intelligent and Robotic Systems, 2001. Volume 31, Issue 1, pp. 91-103

[11] J.T. Connor, R.D. Martin, and L.E. Atlas, *Recurrent neural networks and robust time series prediction*, IEEE Transactions on Neural Networks, Mar 1994. Volume 5, Issue 2, pp. 240 - 254

[12] Johan Bollen1, Huina Mao1, and Xiao-Jun Zeng, *Twitter mood predicts the stock market*, arXiv:1010.3003 [cs.CE]

[13] A. Hernandez, *Model calibration with neural networks*, Risk, June 2017

[14] M. Klos and Z. Waszczyszyn, *Modal analysis and modified cascade neural networks in identification of geometrical parameters of circular arches*, Computers and Structures, 2011, 89:581589

[15] D. Novák and D. Lehký, *ANN inverse analysis based on stochastic small-sample training set simulation*, Engineering Applications of Artificial Intelligence, 2006, 19:731740

[16] K. Zaw, G. R. Liu, B. Deng, and K. B. C. Tan, *Rapid identification of elastic modulus of the interface tissue on dental implants surfaces using reduced-basis method and a neural network*, Journal of Biomechanics, 2009, 42:634641

[17] L. Zhang, L. Li, H. Ju, and B. Zhu, *Inverse identification of interfacial heat transfer coefficient between the casting and metal mold using neural network*, Energy Conversion and Management, 2010, 51:18981904

[18] T. Mare, E. Janouchová, and A. Kuerová, *Artificial neural networks in calibration of nonlinear mechanical models*, arXiv:1502.01380 [cs.NE]

[19] D. Brigo and F. Mercurio, *Interest Rate Models - Theory and Practice: With Smile, Inflation and Credit*, Springer Finance

[20] K. He, X Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs.CV]

[21] S. Ioffe, C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, arXiv:1502.03167 [cs.LG]

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research, 2014, 15, 1929-1958