Distributed Calibration of Option Pricing Models with Multiple Contracts Written on Different Underlying Assets

April 11, 2017

Juho Kanniainen[a,*], Marko Koskinen[b]

[a] *Laboratory of Industrial and Information Management, Tampere University of Technology, P.O. Box 541, FI-33101 Tampere, Finland*

[b] *Techila Technologies Ltd, Itsenäisyydenkatu 2, FI-33100 Tampere, Finland*

Abstract

Think about a situation, where a financial institution has multiple option positions, each written on a different underlying asset, and the unexpected arrival of market-wide news shakes the markets. In the case of such a market-wide news arrival, all the volatility models on different underlyings must be immediately re-calibrated for robust option pricing and hedging. Unfortunately, the calibration of models using data on multiple underlying securities can take too long, especially, if advanced non-affine models without analytical solutions are used. This work demonstrates how multiple independent calibration tasks (on different underlyings) can be efficiently accelerated by Techila, reducing the length of the wall-clock computation time from 7 hours to a few minutes. Option pricing and calibration codes are made open for the financial community.

1. Introduction

Financial institutions can be exposed to the dynamics of hundreds of securities via derivative contracts, which requires robust hedging strategies with well-calibrated volatility models. If everything goes smoothly without major market-wide news, the models can be re-calibrated during nights. However, there can be important macro announcements or other market-wide news arrivals during a trading day that affect the price processes of securities over different asset classes, in which case one should instantly re-calibrate models for all the underlying assets. The calibration of models can take a long time, especially if state-of-the-art non-affine volatility models are used. The extant literature provides strong evidence that many popular variance models with closed-form solutions for option prices, such as (Heston, 1993), are outperformed by non-affine models that, unfortunately, are often non-tractable (Christoffersen et al., 2010; Kaeck and Alexander, 2012; Kanniainen et al., 2014; Yang and Kanniainen, 2016). In order to make the use of these these modern non-tractable models feasible for financial institutions, increased computationally challenged should be solved in an efficient way.

This work elaborates how the computation time can be speeded up by distributed computing with Techila in a situation where there are several underlying assets for which a model(s) should be calibrated. Particularly, state-of-the-art non-affine volatility models are calibrated by minimizing the squared differences between market and model implied volatility surfaces with Monte Carlo methods.

*Corresponding author

Email addresses: juho.kanniainen@tut.fi (Juho Kanniainen), marko.koskinen@techilatechnologies.com (Marko Koskinen)

URL: www.techilatechnologies.com (Marko Koskinen)

2. Model

We use a model that can be seen as a non-affine generalization of Heston model. Let $\{Y_t; t \geq 0\}$ denote the continuously compounded returns on an underlying asset (e.g. the S&P 500 index) and $\{V_t; t \geq 0\}$ denote the instantaneous squared volatility of the returns. Under the risk-neutral measure Q, we assume that the return and volatility dynamics of the underlying is described by the following stochastic differential equations:

$$\begin{aligned} dY_t &= \left(r - \frac{1}{2}V_t\right)dt + \sqrt{V_t}dW_{Y,t}, \\ dV_t &= \kappa\left(\theta - V_t\right)dt + \xi V_t^{\gamma}\left(\rho dW_{Y,t} + \sqrt{1-\rho^2}dW_{V,t}\right). \end{aligned} \tag{1}$$

W_Y and W_v are mutually independent Brownian motions and $Y_0 = y_0 > 0$ and $V_0 = v_0 > 0$. Here κ is the speed of variance mean revision, θ long-run variance, ξ the volatility of variance, and ρ the correlation between returns and volatility. Additionally, γ is the coefficient to make a model for affine ($\gamma = 0.5$) or non-affine (otherwise). Finally, r is the risk-free interest rate, assumed to be 0.1%.

3. Monte-Carlo Pricing

We use three variance reduction methods to accurate the Monte-Carlo pricing of options. First, Black-Scholes price serves as a control variate (see e.g. Glasserman, 2013, Ch. 4.1). Second, antithetic variates (see e.g. Glasserman, 2013, Ch. 4.2) are used for both return and volatility processes. Third, we implemented Empirical Martingale Simulation method, introduced in Duan and Simonato (1998).

The Monte-Carlo simulation is implemented efficiently so that one can compute a cross-section of options with different strike prices and maturity times by one run, and therefore, the size of volatility surface does not essentially affect the computation time. The most important factor is maturity time; if fact, the paths are simulated according to the longest maturity time, and therefore, if a single option contract with a long maturity time can increase the computational time considerably.

4. Calibration

As Broadie et al. (2007) argue, minimizing the error between model and market option dollar-prices places a greater weight on expensive in-the-money and long-maturity options, whereas the implied volatility metric provides an intuitive weighting of options across strikes and maturities. Consequently, we minimize the error between implied volatilities rather than dollar-prices:[1]

$$\text{IVRMSE}(\Theta, v_0) = 100 \times \sqrt{\frac{1}{N}\sum_i \left(IV(f_t(\Theta, v_0)) - IV(\hat{f}_{i,t})\right)^2}, \tag{2}$$

which is minimized with respect to the structural parameters $\Theta = \{\kappa, \theta, \xi, \rho, \gamma\}$ and spot volatility v_0. Here f_i is the price of the ith option given by the model and $\hat{f}_i$ is the corresponding price of the option observed in the market data. Moreover, N is the number of option contracts in the sample.

We use Matlab's `fminsearch` function, provided in optimization toolbox, to minimize the pricing error (Eq. 2), which uses derivative-free Nelder-Mead simplex algorithm (Lagarias et al., 1998). Because the method itself is unconstrained, we crafted constrains in such a way that the loss function (Eq. 2) gets very high values (1e10) if the parameters or spot variance are out of the given boundaries.

[1] Implied volatility errors, on the other hand, can be approximated by scaling dollar prices by (Black, 1976) vegas without extensive computations (see Carr and Wu, 2007; Trolle and Schwart, 2009; Kanniainen et al., 2014).

5. Data and Settings

Instead of using empirical data, we calibrate the model using simulated data so that actual values are know and the accuracy of the parameter estimates can be measured. In the generation of the data sets, 200,000 Monte-Carlo iterations were used to generate the data sets.[2] Additionally, "actual" implied volatilities are computed for the all the combinations of the following maturities and strikes: 3, 6, 9, 12, 18, 24, 36, 60, 120, and 240 months and the $K/S_0 = \{0.5, 0.6, \ldots, 1.5\}$, where K is the strike price and S_0 the current price of the underlying asset.

Assume that a financial institution has m implied volatilty surfaces (for m underlying asset) for which the models should be calibrated. The parameter values used to generate implied volatility surfaces are then computed as

$$x = \min(\max(\bar{x}(1 + \alpha\epsilon), x_{min}), x_{max}), \tag{3}$$

where x refers to a given parameter, epsilon is a standard normal variable, and $\alpha > 0$. For the data set attached, we used $\alpha = 0.4$, which creates quite different data sets. Additionally, we used the following mean values and minimum and maximum values for the parameters provided in Table 1. Figure 1 shows two different volatility surfaces generated by these settings.

Table 1: The mean, minimum and maximum values of parameters used in data generation.

	κ	θ	ξ	ρ	γ	v_0
Mean	2	0.25^2	0.8	-0.8	0.7	0.3^2
Min	0.5	0.01^2	0.0025	-1	0	0.01^2
Max	6	1	2	1	1	1

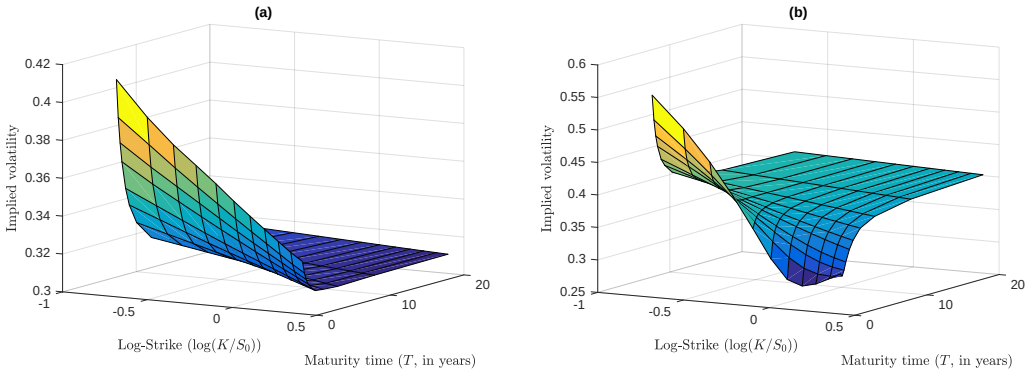

Figure 1: Two volatility surfaces generated with different random parameters. The plot (a) on left hand side is based on parameters $\kappa = 2.0012, \theta = 0.0984, \xi = 0.4188, \rho = -0.7189, \gamma = 0.9405, v_0 = 0.1352$ and the plot (b) on right hand side is based on parameters $\kappa = 4.0191, \theta = 0.0775, \xi = 1.0235, \rho = -0.5455, \gamma = 0.1720, v_0 = 0.0952$.

The starting values of parameters used in optimization are determined as follows:

$$x_0 = x_{\text{true}} \exp(\eta\varepsilon - \eta/2),$$

where ε is a standard normal random variable and $\eta = 0.1$. Additionally, x_{true} is given by Eq (3). That is, in order to use "good" initial values, we determine their values to be in the neighborhood of the true values.

[2] We decided to use a substantially large number of iterations in data set generation to minimize Monte-Carlo errors with the data used for model calibrations.

6. Infrastructure

Computations were performed in Microsoft Azure cloud using Techila Distributed Computing Engine. The computational capacity consisted of 25 D3v2 virtual machines, which each have 4 CPU Cores. This means that the total amount of CPU cores used in the computations was 100. Figure 2 illustrates the Techila Distributed Computing Engine architecture when using Microsoft Azure.

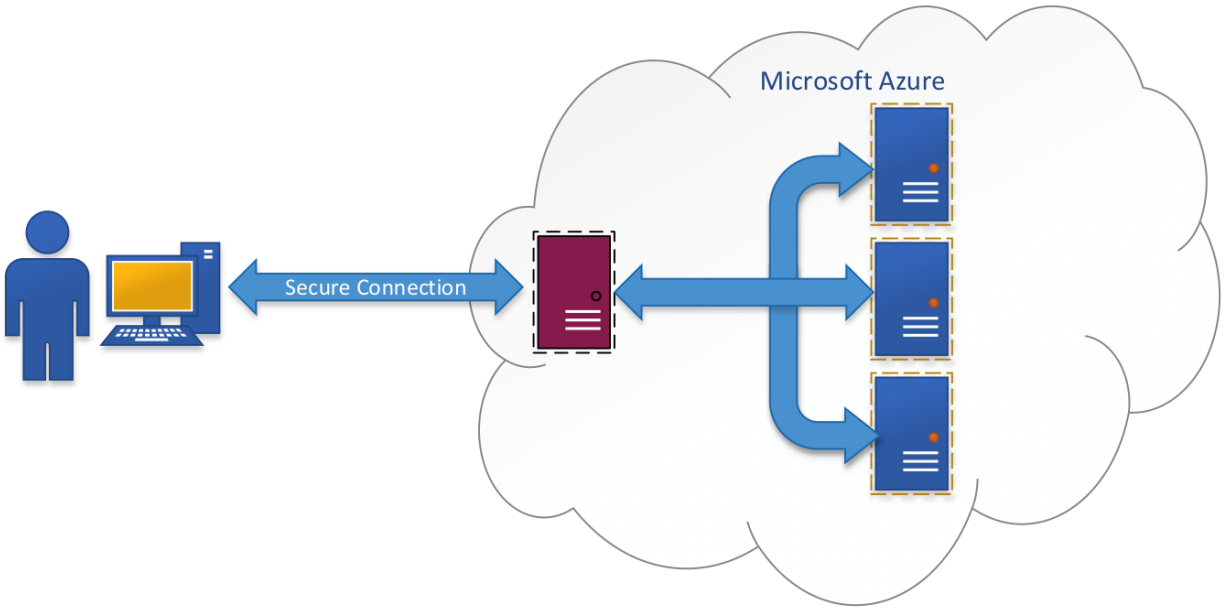

Figure 2: The Techila Distributed Computing Engine architecture when using Microsoft Azure.

7. Results

Figure 3 shows the option pricing errors (IVRMSE) before and after calibration procedures for all the 100 assets. The average of initial IVRMSEs over 100 assets is 1.072 and the average of final IVRMSE is 0.601. Figure 4 visually demonstrates three surfaces for one asset: A non-transparent surface represents (market) data for which the model is calibrated, a transparent surface represents the model with initial parameters, and a green surface is a calibrated model with optimized parameters.

The CPU time used to calibrate the mode for 100 volatility surfaces was **7 h 50 m 10 s**, which is a rough approximation how long it would have taken to perform the computations on a single CPU core machine with similar hardware specifications. If we would take into account the overheads inherently related to distributed computing, the actual computational time would be slightly smaller, but still non-trivial. By using 100 CPU cores in Techila Distributed Computing Engine, the computations were completed in **5 minutes 19 seconds**.

Table 2: CPU and wall clock times with different number of cores.

Amount of CPU Cores	Threading	CPUs per Job	CPU Time	Wall clock time
100	Single threaded	1	7h 50m 10s	5 m 19 s
200	Multithreaded	2	8h 13m 26s	3 m 19s
400	Multithreaded	4	9 h 46 m 39 s	2 m 11 s
800	Multithreaded	8	12 h 33 m 12 s	1 m 55 s

As the computational workload contained vectorized operations, the wall clock time could be reduced further by utilizing MATLAB's own multithreading functionality in the Techila Distributed

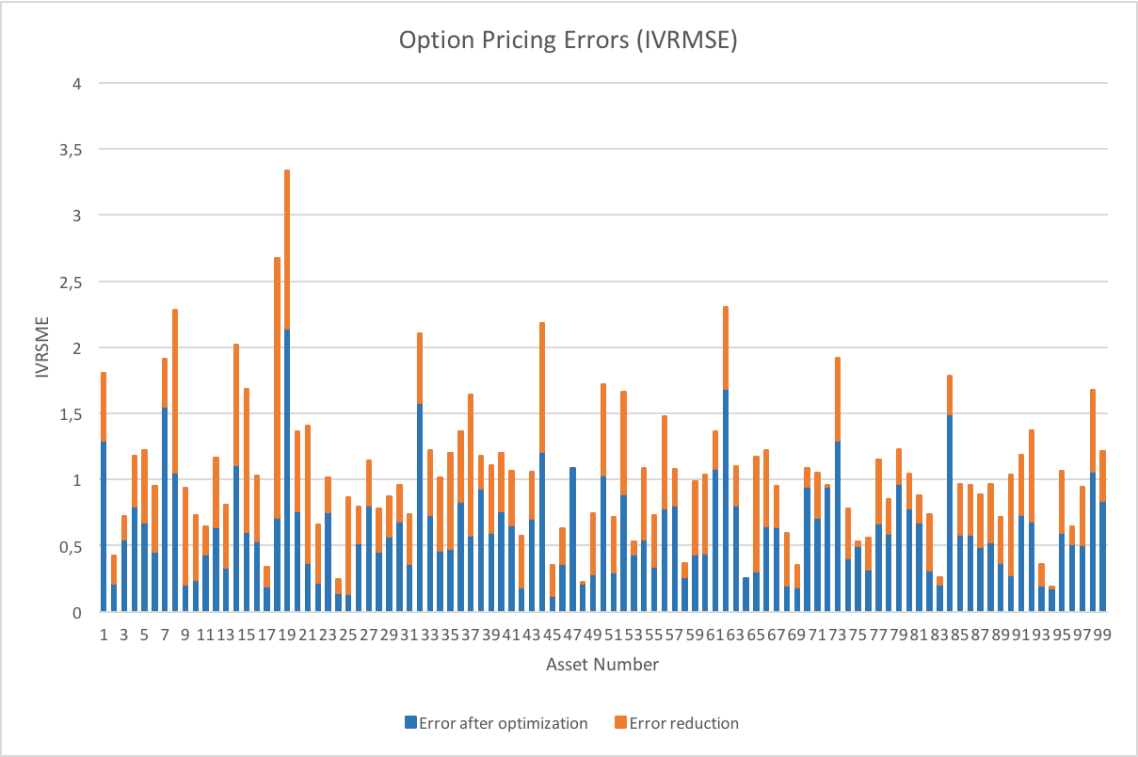

Figure 3: Option pricing errors (IVRMSE) before and after calibration procedures for all the 100 assets. The average of initial IVRMSEs over 100 assets is 1.072 and the average of final IVRMSE is 0.601.

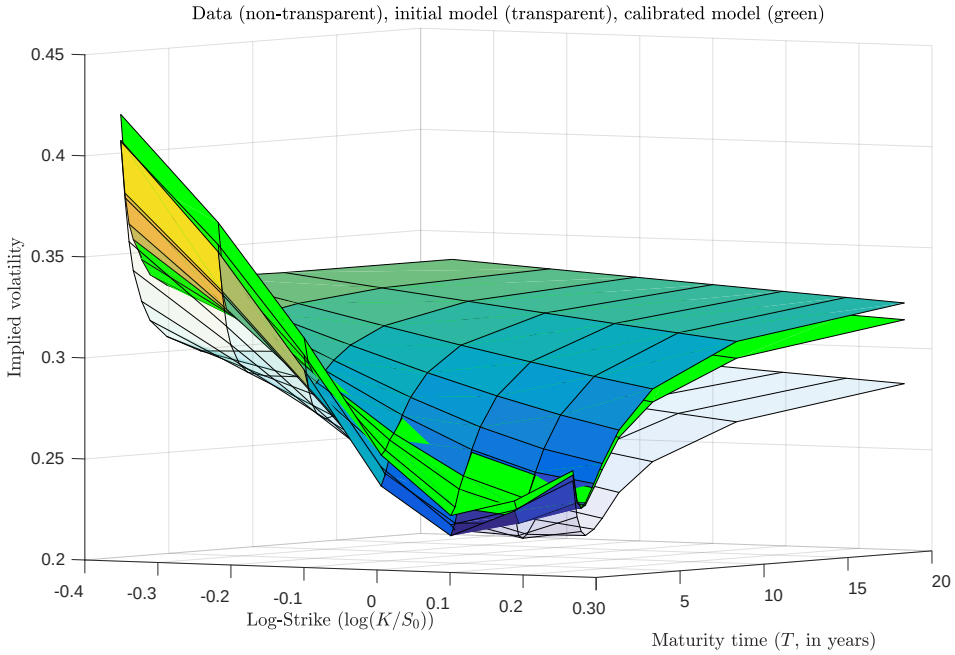

Figure 4: Demonstration of calibrated implied volatility surface for one asset. A non-transparent surface represents (market) data, a non-transpanent surface represents the model with initial parameters, and a green surface a calibrated model with optimized parameters.

Computational Engine. To measure the effect of these multithreading operations on performance, three additional tests were performed in computing environments consisting of 200, 400 and 800 CPU cores, which allowed each Job to utilize 2, 4 or 8 CPU cores respectively. These results are provided in Table 2 and demonstrated in Figure 5. As expected, this reduced the wall clock time but also increased the total CPU time due to increased overheads related to the multithreading operations.

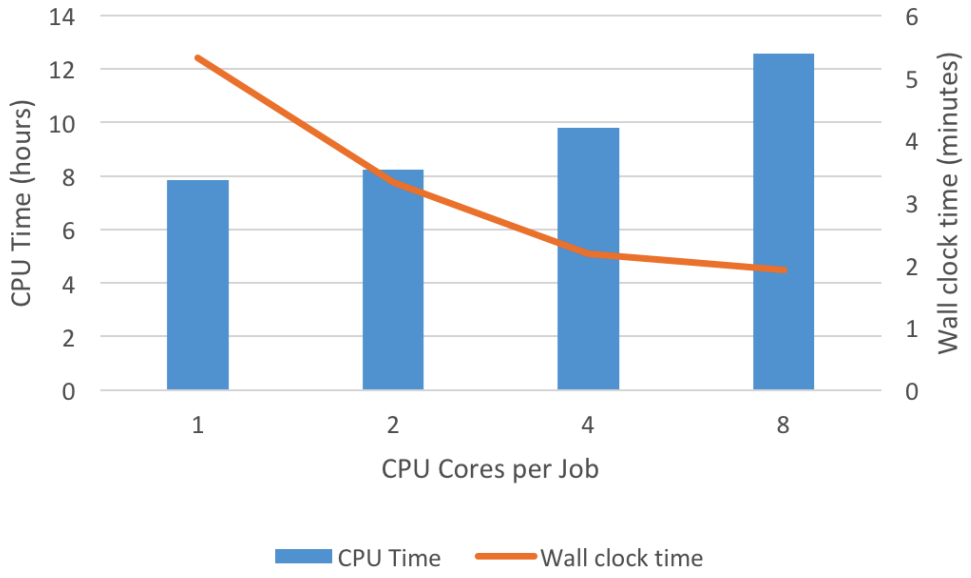

Figure 5: Effect of multithreading on performance.

References

Black, F., 1976. The pricing of commodity contracts. Journal of Financial Economics 3, 167–179.

Broadie, M., Chernov, M., Johannes, M., 2007. Model specification and risk premia: Evidence from futures options. Journal of Finance 62, 1453–1490.

Carr, P., Wu, L., 2007. Stochastic skew in currency options. Journal of Financial Economics 86, 213–247.

Christoffersen, P., Jacobs, K., Mimouni, K., 2010. Volatility dynamics for the S&P500: evidence from realized volatility, daily returns, and option prices. Review of Financial Studies 23, 3141–3189.

Duan, J.C., Simonato, J.G., 1998. Empirical martingale simulation for asset prices. Management Science 44, 1218–1233.

Glasserman, P., 2013. Monte Carlo methods in financial engineering. volume 53. Springer Science & Business Media.

Heston, S.L., 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. Review of financial studies 6, 327–343.

Kaeck, A., Alexander, C., 2012. Volatility dynamics for the S&P 500: Further evidence from non-affine, multi-factor jump diffusions. Journal of Banking & Finance 36, 3110–3121.

Kanniainen, J., Lin, B., Yang, H., 2014. Estimating and using GARCH models with VIX data for option valuation. Journal of Banking & Finance 43, 200–211.

Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E., 1998. Convergence properties of the nelder–mead simplex method in low dimensions. SIAM Journal on optimization 9, 112–147.

Müller, J., Kanniainen, J., Piché, R., 2013. Calibration of garch models using concurrent accelerated random search. Applied Mathematics and Computation 221, 522–534.

Trolle, A.B., Schwart, E.S., 2009. Unspanned stochastic volatility and the pricing of commodity derivatives. Review of Financial Studies 22, 4423–4461.

Yang, H., Kanniainen, J., 2016. Jump and volatility dynamics for the S&P 500: Evidence for infinite-activity jumps with non-affine volatility dynamics from stock and option markets. forthcoming in Review of Finance doi:10.1093/rof/rfw001.