The calibration conundrum

Laura Ballotta[a]

23rd August 2024

When dealing with pricing of derivatives contracts, no matter the nature of the underlying (equity, interest rates, FX, commodity,...) or the payoff structure, a Quant needs a trusted model, and a meaningful and consistent set of model parameters. The latter is usually obtained from market data, i.e. by identifying the parameters which are most likely given by the prices of traded (liquid) options. This procedure, known as calibration is essentially a minimization problem involving distances, or errors, between the market data and the counterparties generated by the model, and it comes with many questions and issues.

Q. Which metric do we use to build the objective function?

A. The mean squared error (MSE) is a valid metric as in an arbitrage free market the price is expressed by a conditional expectation (see Hastie et al., 2017, for example). Monotone transformations, such as the root mean squared error (RMSE) for example, can be useful too.

Q. Do we use prices or implied volatilities in the construction of the MSE?

A. Implied volatilities should be preferred to ensure comparability, even though this choice requires a double step: first a (fast and accurate) pricing routine, and then the 'inversion' of the Black and Scholes (1973) formula to recover the volatilities. Thus, the procedure could be computationally lengthy, as argued to some extent by Christoffersen et al. (2009). A viable compromise could be the rescaling of the prices by the option vega to obtain a first order approximation (in Taylor sense) of the implied volatility (see Carr and Wu, 2004; Christoffersen et al., 2009, amongst others).

Q. What data do we use for calibration?

A. Liquid option prices, that is out of the money (OTM) puts and calls, where the moneyness is fixed on the basis of the underlying asset spot price or the 'implied' forward price.

Q. Do we obtain a unique minimum?

A. No: the calibration problem is an ill-posed minimization problem, as the objective function is usually non-convex. This originates many local minima due to the presence of long 'valleys' with minimal difference in the gradient of the objective function (see Cont and Tankov, 2004; Cont, 2010, for example).

[a]Laura Ballotta is a Professor of Mathematical Finance at the Faculty of Finance, Bayes Business School (formerly Cass), City St George's, University of London, UK.
email: L.Ballotta@city.ac.uk.

This is where the Pandora's box opens: there might be for example a significant dependence of the solution to the minimization problem on the initial condition due to the already mentioned lack of convexity of the objective function. This lack of convexity can also compromise the stability of the optimal parameter set from one day to the next: after a small change in market prices, the optimal solution might indeed move considerably along the valley characterizing the objective function. An extensive discussion of these points together with possible solutions can be found in Cont and Tankov (2004) for example.

Here I want to focus on a different but strictly connected aspect of the calibration procedure. Suppose we can solve all the above mentioned problems in a satisfactory way. We still have to run a minimization routine and therefore we still have to choose a suitable algorithm from our programming platform of choice, which we implicitly trust to lead us to the promised land: the optimal parameter set of our chosen model.

Is this trust well posed?

To illustrate the point of this question, I use the following controlled experiment.

In first place, I generate from my chosen model a pseudo market dataset. Thus, for illustration purposes (and for simplicity), I choose the CGMY process of Carr et al. (2002) (also reviewed in Ballotta, 2023) with characteristic function $\phi_{L_t}(u) = exp(\varphi(u)t)$, for

$$\varphi(u) = C\Gamma(-Y)\left((G + \mathrm{i}u)^Y - G^Y + (M - \mathrm{i}u)^Y - M^Y\right),$$

and parameters $C = 0.1$, $G = 2.0$, $M = 3.5$, $Y = 1.5$. Thus, for a spot price S, the price of the underlying asset at time $t > 0$ is $S_t = S \times exp(rt + X_t)$ for $X_t = -\varphi(-\mathrm{i})t + L_t$. With this model, I generate prices of 30 days to maturity (i.e. $T = 30/365$) OTM puts and calls spanning a range of moneyness $K/S \in [0.75, 1.2]$, where K denotes the strike price; subsequently, I use the Black-Scholes formula to recover the implied volatility smile. For pricing, I use the Fourier method of Eberlein et al. (2010) so that the option price is

$$O(K) = e^{-rT}\frac{e^{-Rs}}{\pi}\int_0^\infty \Re\left(e^{-\mathrm{i}us}\phi_{X_T}(u - \mathrm{i}R)\frac{K^{1-R-\mathrm{i}u}}{(-R - \mathrm{i}u)(1 - R - \mathrm{i}u)}\right) du,$$

for $s = -\ln S$ and a dampening parameter R to be set $R > 1$ for call options and $R < 0$ for put options. Integrals are computed in Matlab with the `integral` function, fully vectorized for speed (see Ballotta, 2022, as well).

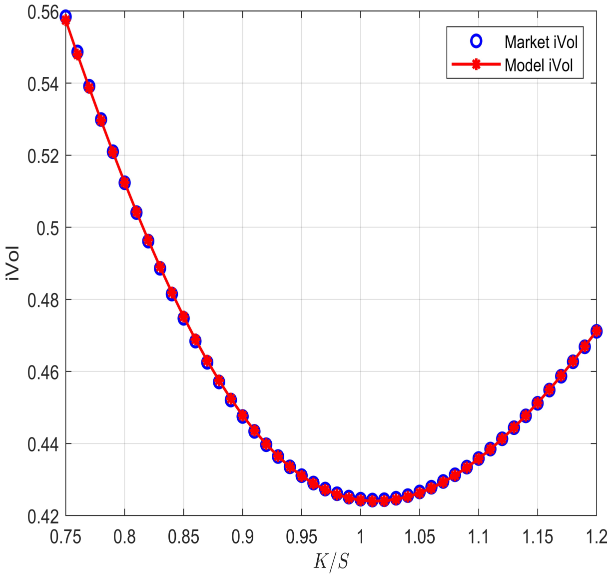
In second place, I assume that I am able to detect the risk driver of the market, i.e. the CGMY process, but not its parameters. Then, I fix an initial set of parameters as input to the minimization algorithm for the calibration. In details, the objective function of the calibration problem is given by

$$F(\vartheta) = \sum_{k=1}^{N}\left(iVol^{mod}\left(K_k;\vartheta\right) - iVol^{mkt}\left(K_k\right)\right)^2,$$

with ϑ denoting the model parameter set, i.e. $\vartheta = (C, G, M, Y)$ for this specific example. $iVol^{mod}\left(K_k;\vartheta\right)$ denotes the Black-Scholes implied volatility extracted from the model price of a contract with strike K_k, $k = 1, \dots, N$, N being the total number of contracts, with parameter set ϑ (we remind that in this simple exercise maturity is kept constant at 30 days). $iVol^{mkt}\left(K_k\right)$ is the market implied volatility of the same contract.

As my platform of choice (Matlab) conveniently has a dedicated routine (`lsqnonlin`) for least

Figure 1: Calibration fit of the CGMY model to the pseudo market dataset achieved with Matlab `lsqnonlin`. Optimal parameter set reported in Table 1. Initial condition: $C = 0.6$; $G = 5.8$; $M = 18.3$; $Y = 1.8$.



squares problems such as the calibration discussed here, I use it as my first candidate to attempt obtaining the optimal parameter set. The routine uses the 'trust-region-reflective' algorithm based on the approximation of the objective function with a simpler function which reasonably reflects the behaviour of the objective function in a neighbourhood around the given point. This works also in presence of bound-constrained problems, like mine as I need to ensure reasonable lower and upper bounds for the parameters, i.e. $C, G, M > 0$ and $Y \in (0, 2)$.

The fit of the implied volatility is shown in Figure 1, and the quality looks remarkable. However, the third row of Table 1 shows that the parameters are not very close to the 'true' ones from which I started my experiment (second row of Table 1). For such a simple calibration problem, one might even argue that the error metrics of choice, i.e.

$$RMSE = \sqrt{\frac{F(\vartheta)}{N}}$$

and

$$APE = \frac{1}{N} \sum_{k=1}^{N} \left| \frac{O^{mod}(K_k; \vartheta) - O^{mkt}(K_k)}{O^{mkt}(K_k)} \right|,$$

as well as the resulting value of the objective function are not particularly impressive.

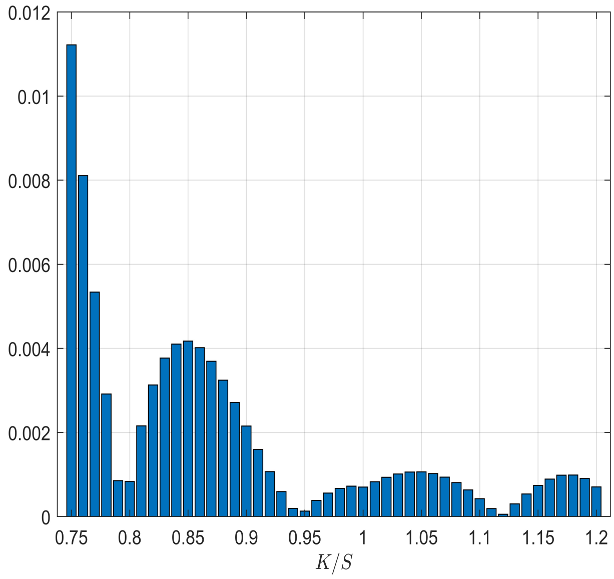
Inspired by Mrázek et al. (2016), I use the same set of initial parameters and bounds to test other optimization algorithms available in Matlab such as `fmincon` - a local optimizer based on gradient descent - and a few global optimizers such as Genetic Algorithm, Global Search and Multi Start. As Matlab allows to customize certain features of these algorithms, in addition to the default setting, I also consider a couple of 'customizations' based on the `fmincon` routine. The choice is motivated by the results in the fourth row of Table 1: this routine returns a parameter set which is very close to the original one, and performs very strongly in terms of fit as shown also in Figure 2 illustrating the absolute percentage errors (APEs) for each price in the sample.

The Genetic Algorithm (`ga`) repeatedly modifies a population of individual solutions by ran-

Table 1: Optimal parameter sets obtained for the calibration of the CGMY model to the pseudo market dataset. Routines of interest: non linear least squares (`lsqnonlin`); constrained minimization (`fmincon`); Genetic Algorithm (`ga`); Global Search (`gs`); Multi Start (`ms`). Initial condition (kept fixed for all algorithms): $C = 0.6$; $G = 5.8$; $M = 18.3$; $Y = 1.8$.

	C	G	M	Y	O.F.	F. Evals.	RMSE iVol	APE Price	CPU (sec)
Original parameters	0.1	2	3.5	1.5	-	-	-	-	-
`lsqnonlin`	0.118949	2.295202	3.820629	1.458416	4.1465E-06	270	3.0023E-04	1.5474E-03	3.46
`fmincon`	0.100203	2.003428	3.503760	1.499521	5.8969E-10	970	3.5804E-06	1.8291E-05	5.97
`ga`	0.298868	3.945294	5.585775	1.220572	1.3802E-04	18853	3.5804E-06	9.0510E-03	270.86
`ga` hybrid (`fmincon`)	0.100297	2.004999	3.505465	1.499301	1.2518E-09	19353	5.2166E-06	2.6913E-05	265.09
`gs`	0.100161	2.002706	3.502956	1.499622	3.6686E-10	66871	2.8240E-06	1.4589E-05	281.65
`ms` (`fmincon`)	0.100154	2.002600	3.502847	1.499636	3.3900E-10	58526	2.7147E-06	1.3949E-05	394.12
`ms` (`lsqnonlin`)	0.108676	2.140973	3.653332	1.480156	1.0818E-06	15755	1.5335E-04	7.3013E-04	171.65

Figure 2: Absolute percentage error (APE) for the price of each contract in the pseudo market set obtained by calibrating the CGMY model with the gradient-descent routine `fmincon`. Initial condition: $C = 0.6$; $G = 5.8$; $M = 18.3$; $Y = 1.8$. APE expressed in percentage.



domly selecting the individuals which produce the next generation, pretty much mimicking the process driving biological evolution. To keep the CPU time under control, I select an initial population of only 20 combinations of parameters generated randomly from the exponential distribution with unit parameter. The algorithm stops when either the maximum number of generations is met, or the relative change in the best fit function value is less than a pre-specified tolerance, which I fix at `1e-12`, following Mrázek et al. (2016). Precision can be increased by enlarging the initial population, at the cost though of increased CPU time. The results in the fifth row of Table 1 show that overall the Genetic Algorithm does not perform significantly better than the non linear least squares routine in terms of APE; the optimal parameters though are much farther from the true ones.

I also test a hybrid version of Genetic Algorithm which continues the optimization after `ga` terminates, using an optimizer of choice, which in this case is `fmincon`. This set up brings a noticeable improvement as shown in the sixth row of Table 1.

Global Search (`gs`) starts `fmincon` from multiple start points, which are generated by a scatter-search mechanism within our finite bounds. These points are analysed and rejected if unlikely to improve the best local minimum found so far. Results in row 7 of Table 1 show that Global Search performs at par with the constrained minimization routine in both recovering the exact parameters and minimizing the error. The CPU cost is quite different though.

Very similar performance is also achieved by the Multi Start (`ms`) algorithm when the local minimizer is set to the gradient-descent method of `fmincon` as well. Multi Start operates very similarly to Global Search except for two aspects: the start points are randomly generated from the uniform distribution within the pre-specified bounds, and the local solver can be defined by the user. Consequently, I also test a setting in which Multi Start uses the least squares optimizer `lsqnonlin`. As shown in the final row of Table 1, this pairing improves the performance of the least square routine but only in terms of distance from the true parameters.

Other choices of global optimization routines are of course possible. But the point of this controlled experiment is to raise awareness about the trust we grant to our precious algorithm for optimization when we build the 'calibrator'. In the long valleys generated by the non-convex objective function, some routines stand a better chance of catching a local minimum close to the global one and these routines might not be global optimizers.

References

Ballotta, L., 2022. Powering up Fourier valuation to any dimension. Wilmott 2022, 68–71.

Ballotta, L., 2023. Demystifying generic beliefs on jump models. Wilmott 2023, 70–73.

Black, F., Scholes, M., 1973. The pricing of options and corporate liabilities. Journal of Political Economy 81, 637–654.

Carr, P., Geman, H., Madan, D.B., Yor, M., 2002. The fine structure of asset returns: An empirical investigation. Journal of Business 75, 305–332.

Carr, P., Wu, L., 2004. Time-changed Lévy processes and option pricing. Journal of Financial Economics 71, 113–141.

Christoffersen, P., Heston, S., Jacobs, K., 2009. The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well. Management Science 55, 1914–1932.

Cont, R., 2010. Model calibration, in: Encyclopedia of Quantitative Finance. Wiley, pp. 1–10.

Cont, R., Tankov, P., 2004. Non-parametric calibration of jump–diffusion option pricing models. Journal of Computational Finance 7, 1–49.

Eberlein, E., Glau, K., Papapantoleon, A., 2010. Analysis of Fourier transform valuation formulas and applications. Applied Mathematical Finance 17, 211–240.

Hastie, T., Tibshirani, R., Friedman, J., 2017. The Elements of Statistical Learning. Springer Series in Statistics, Springer. Second edition.

Mrázek, M., Pospíšil, J., Sobotka, T., 2016. On calibration of stochastic and fractional stochastic volatility models. European Journal of Operational Research 254, 1036–1046.